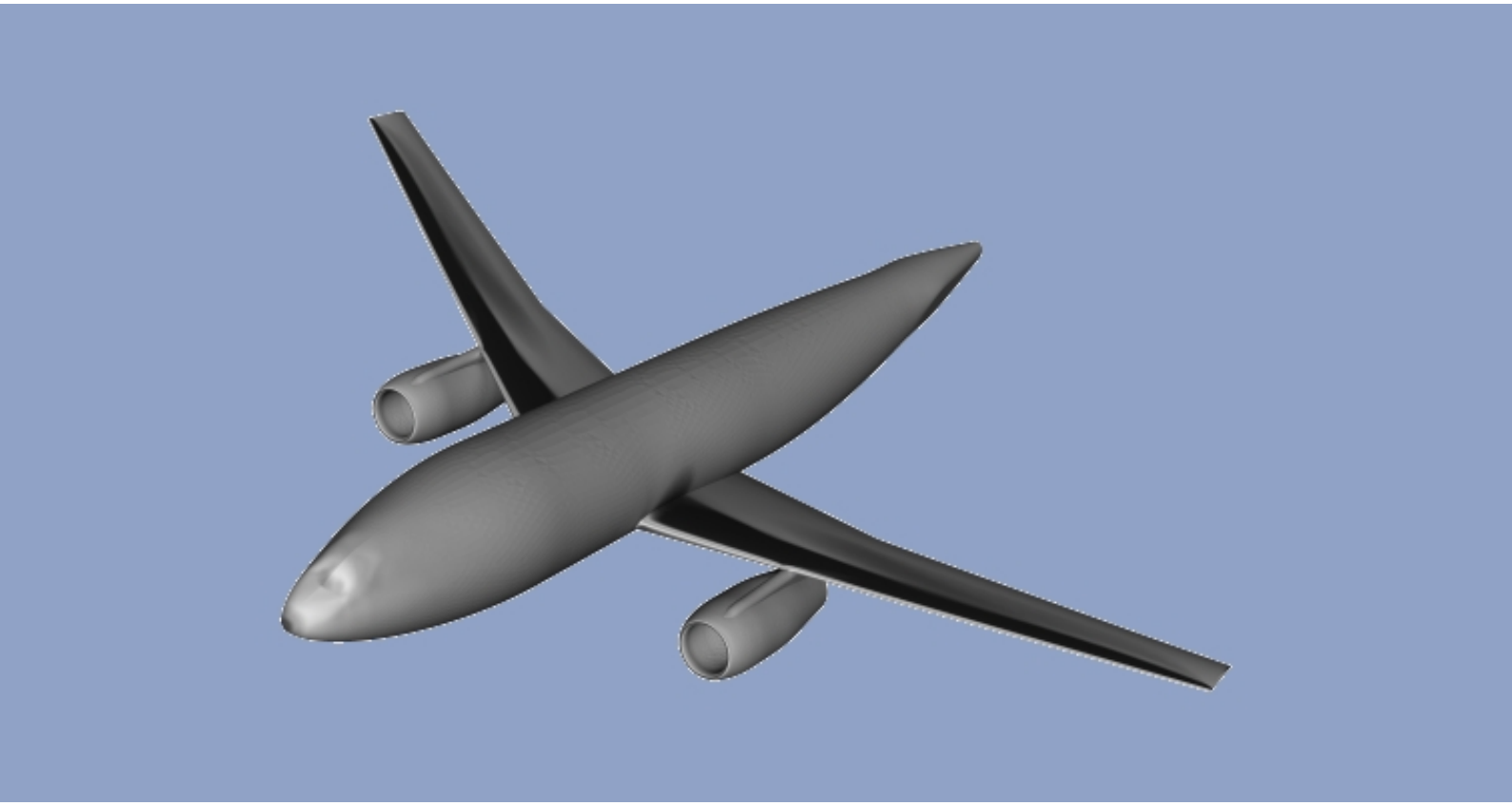
FOI

Edge
User Guide



Edge

Defence and Security, Systems and Technology
FOI dnr 03-2870

March 2007
Issue 4.1.0

Edge
User Guide

FOI dnr 03-2870
ISSN-1650-1942

March 2007
Issue 4.1.0

Contents

1 Introduction **11**

2 Installation **13**

2.1 Introduction 13
2.2 Prerequisites 13
2.2.1 Required 13
2.2.2 Optional 13
2.3 Directory Structure 14
2.4 Building Edge 15
2.4.1 Configuring Using the Configure Script 15
2.4.2 Compiling 17
2.4.3 Installing 17
2.4.4 Build Java Based GUI (optional). 17
2.4.5 Testing your Installation 18

3 Running Edge **19**

3.1 Steps to Take to Run Edge 19
3.1.1 Setting Up an Input File 19
3.1.2 Specifying Boundary Conditions 20
3.1.3 Running the Preprocessor 21
3.1.4 Running the Flow Solver 22
3.1.5 Postprocessing 23
3.1.6 Running Edge in Parallel 23
3.1.7 Running the Grid Adaption 23
3.1.8 Running Xedge 25
3.2 List of all Programs in Edge 25
3.3 The FFA Dataset/Fileformat 25
3.4 Files Generated/Required by Edge 29
3.4.1 The Input Grid File **Edge.bmsh** 29
3.4.2 The Boundary Condition File **Edge.aboc** 30
3.4.3 The Edge File **Edge.bedg** 30

3.4.4 The Solution File **Edge.bout** (**Edge.bini**) . . . 31
3.4.5 The Sample Solution Files **Edge_timestp.bout** . . . 31
3.4.6 The Solution File for Post Processing **Post.bout** . . . 34
3.4.7 The Residual File **Edge.bres** . . . 34
3.4.8 The Sample Solution File **Edge_adjoint.bout** . . . 34
3.5 Hints and Recommendations . . . 36
3.5.1 Specifying the Free Stream Viscosity and Reynolds Number . . . 36
3.5.2 Turbulent Flow Calculations . . . 36
3.5.3 Convergence Problems . . . 39
3.5.4 Time Accurate Simulation and Inner Iterations . . . 39
3.6 Boundary Conditions in Edge . . . 40
3.6.1 Connectivity Boundary Conditions . . . 41
3.6.2 Wall Boundary Conditions . . . 42
3.6.3 Symmetry Boundary Conditions . . . 42
3.6.4 External Boundary Conditions . . . 42
3.7 Using the Vortex Generator Model . . . 46
3.7.1 The VG Geometry File **Edge.avg** . . . 47
3.8 Aeroelastics Options and Programs . . . 47
3.8.1 Overview of the Aeroelastics “ae” Programs . . . 48
3.8.2 Program **aexbset** – Moving Surface Definition . . . 49
3.8.3 Program **aedbrot** – Rigid Rotation Surface Movement . . . 51
3.8.4 Program **aedbelast** – Modal Elastic Surface Movement . . . 51
3.8.5 Program **aetribend** – Static Mesh Deformation . . . 54
3.8.6 Program **aeputpert** – Modal Perturbation Fields . . . 55
3.8.7 The **mottab** Signal Tabulation System . . . 56
3.9 Edge Modal Coupled Solutions . . . 58
3.9.1 Structural Model and Modal Parameters . . . 58
3.9.2 Coupled Simulation Procedures . . . 59
3.10 Edge Adjoint Calculation for Inviscid Flow . . . 61
3.10.1 Serial Computation . . . 62
3.10.2 Parallel Computation . . . 63
3.11 The Meshdeform Utility . . . 63

A Appendix – Input Variables **67**
A.1 General Parameters for Several Programs . . . 67
A.2 Preprocessor . . . 69
A.3 Flow Solver . . . 70
A.4 Grid Adaption . . . 81
A.5 Adjoint Euler . . . 83

B Appendix – Datasets in FFA-format **85**

B.1 Datasets for Input and Output Data 85

Abstract

This is the users manual for the computational fluid dynamics (CFD) solver **Edge**.

Edge is a parallelized CFD flow solver for 2D/3D viscous/inviscid compressible flow on unstructured grids with arbitrary elements. Edge can be used for both steady state and time accurate calculations including manoeuvres and aeroelastic simulations. It can be used to compute both viscous and inviscid flow problems. It also contains mesh adaption functionality and an inviscid adjoint flow solver.

This users manual describes Edge from a user's point of view. It contains information on how to install and run the Edge solver and related utility programs such as preprocessor, postprocessor, various file-conversion utilities etc. All required input parameters are described as well as the contents of the solution files and other output files.

This document does not contain a complete theoretical formulation of the numerical methods in Edge. Nor does it contain details on implementation or design for software developers.

1 Introduction

This document is part of Edge, a parallel CFD code developed by FOI, see Eliasson (2001) and Edge homepage. Edge is a parallelized CFD flow solver system for solving 2D/3D viscous/inviscid, compressible flow problems on unstructured grids with arbitrary elements. Edge can be used for both steady state and time accurate calculations including manoeuvres and aeroelastic simulation. There is also an inviscid adjoint flow solver that can be used for gradient based shape optimization. There is also functionality for mesh deformation, postprocessing, file format conversion and other utility tasks.

The Edge system comprises five major components:

- **Preprocessor** – constructs the dual mesh and edge based datasets
- **Edge flow solver** – performs the main flow computation
- **Helper programs** – a general "toolbox" for manipulating input and output data
- **Application programs** – programs for shape optimization mesh deformation and aeroelasticity
- **Xedge** – a graphical user interface (GUI) generator for Edge

The flow solver employs an edge-based formulation which uses a node-centered finite-volume technique to solve the governing equations. The control volumes are non-overlapping and are formed by a dual grid, which is computed from the control surfaces for each edge of the primary input mesh. The relationship between the dual and primary input mesh is illustrated in Figure 1. In this example, a set of hexagonal control volumes are constructed from a simple triangular input mesh. In any Edge mesh, all the mesh elements are connected through matching faces. Edge meshes therefore may not contain hanging nodes.

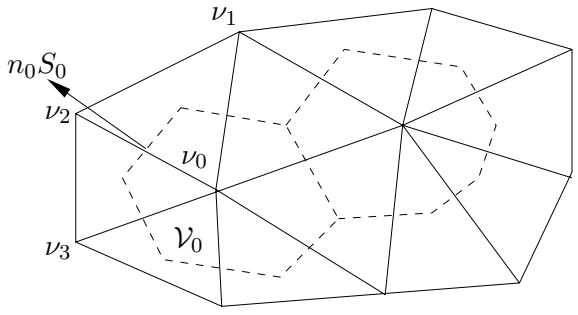


Figure 1. The input grid (*solid*) also denoted triangular grid and its dual grid (*dashed*) forming the control volumes.

In the flow solver, the governing equations are integrated explicitly towards steady state with Runge–Kutta time integration. Convergence is accelerated using agglomeration multigrid and implicit residual smoothing. Time accurate computations can be performed using a semi

implicit, dual time stepping scheme which exploits convergence acceleration technique via a steady state form inner iteration procedure.

Every program in the Edge system can be used via a command-line interface, which is implemented as set of simple Unix shell scripts. However, there is also a platform-independent graphical user interface (GUI) written in Java, which allows configuring a complete Edge computation with on-line help.

The GUI is recommended, especially for new users of Edge, mainly because it greatly simplifies the process of setting up the input for an Edge computation.

This manual is divided into two main parts: a User guide, describing how to install and use Edge, and an Appendix containing reference information such as file formats and definitions of input variables.

2 Installation

2.1 Introduction

Installation from source code involves configuring, compiling and installing. Care has been taken to make the installation process conform with common practice used in other free software distributions for Unix platforms.

This section does not apply for users installing the binary distribution of Edge. The binary distribution is installed simply by unpacking the compressed tar file.

2.2 Prerequisites

2.2.1 Required

To compile and install Edge the following tools are required:

- A **Unix platform** (e.g. Linux, OSF1, Irix, AIX). Any common workstation should work fine. MIMD-type parallel computers (e.g. Linux-clusters) are supported.
- A **Fortran 95 compiler** e.g. Intel ifc 7, Intel ifort 8 or 9, lahey f95. (GNU gfortran 4.1.0 will not work due to some compiler bugs and incompatibility of unformatted I/O).
- **Bash** (Bourne Again Shell) v2.05b or later.
- **GNU make** 3.79.1 or later
  (freely available at http://www.gnu.org/software/make/make.html)
- A **C-compiler**.

2.2.2 Optional

The following tools are optional. If an optional component is not available Edge will be built without the functionality that requires the unavailable package(s):

- **Java 2 Platform Standard Edition v1.4 Runtime Environment**. Required to run Xedge.

Filename	Contents
configure	A script used to automatically configure Edge.
COPYRIGHT	Copyright statements.
INSTALL	Installation instructions.
INSTALL.BINARY	Installation instructions for binary distribution.
Makefile	Top-level makefile.
README	Short description of the directory contents.
rules.mk	Makefile snippet containing generic build-rules.

Table 1. Files in the root directory.

- **Java 2 Platform Standard Edition v1.4 Development Kit**. Required to recompile Xedge.
- An **MPI implementation** (e.g. LAM, MPICH). Required to take advantage of a parallel computer.
- **METIS**. Required to partition the computation problems on parallel computers.
- **NetCDF**. Used for conversion utility to/from NetCDF format files.
- **CGNSlib**. Used for conversion utility to/from CGNS format files.
- **LAPACK** and **BLAS**. Actually not required, a stripped implementation is included with the Edge software distribution, but optionally an external implementation may be used instead. Used by a mesh-adaption utility.
- **UMFPACK** and **AMDLIB**. Used by a mesh-deformation utility.
- **Python 2.4**, **GTK** and **pyGTK**. Required to use the **ffaview.py** utility (FFA-file viewer/editor).
- **Xmgr**. Used by **plotres** to plot convergence curves.

2.3 Directory Structure

The Edge source code is usually supplied as a compressed tar file. This file should be uncompressed and unpacked in a chosen installation directory, for instance using the following Unix command:

```
gunzip edge.tgz | tar xf -
```

After unpacking the Edge tarfile, the files and directories listed in Table 1 and 2 respectively should appear.

The source code directories, listed here are the first level of a hierarchical structure and compilation system based on Makefiles. Configuration settings for the compilation process are determined automatically by running the configure script. Some additional configuration parameters are stored in a set of files in the **config/** directory. The Xedge GUI is compiled separately from the main Edge system. The Matlab programs supplied with Edge are based on the FFA Matlab Toolbox, a special set of Matlab functions for handling the FFA data format (see Section 3.3). This Toolbox is not specific to Edge and so is supplied separately.

Directory	Contents
adaption	Mesh adaption source code.
ae_programs	Aeroelastic utility programs source code.
ae_util	Various aeroelasticity-related source code files.
bin	Some miscellaneous utility scripts, mostly used when building Edge.
config	Makefile snippets for different platforms/compilers.
doc	Documentation files.
examples	Example files for running Edge, including a generic, default input file.
gui	Xedge graphical user interface source code.
lapack	A LAPACK implementation.
matlab	Matlab functions.
preprocessor	Preprocessor source code.
programs	Various utility programs source code.
solver	The flow solver source code.
src_util	Various source code files.
util	Various source code files dealing with the FFA file-format.

Table 2. Directories in the root directory.

2.4 Building Edge

The installation procedure given here is also found in the file **INSTALL** in the root directory. Compiling Edge requires three steps: configuring, building and installing.

2.4.1 Configuring Using the Configure Script

The makefiles named **Makefile** should not be edited. Instead, the script **configure** must be run before building Edge. It configures Edge for building on your specific platform.

The **configure** script is located in the root directory of the source code. It is recommended to prepend the configure command with a path (e.g. **./configure**) in order to avoid mix-up with any other configure script that may exist in your PATH.

The **configure** script requires bash to run. If bash is not in the default location (/bin/bash) the script will fail. In this case you should run "bash configure" instead of "**./configure**".

You may choose to compile Edge in a directory tree separated from the source code tree. This can be convenient when building Edge for multiple platforms from the same source code directory, or from a source code directory to which you have no write-access. To accomplish this, change directory to the build directory and run the configure script from there. E.g.

```
mkdir edge.build
cd edge.build
../edge/configure
```

The following subsections describe the command-line options that can be given to **configure**. In most cases you don't need to specify any option to **configure**, the script then attempts to determine all settings automatically.

2.4.1.1 `--help`

The **configure** script takes several optional parameters, type

```
./configure --help
```

for a complete listing of available options.

2.4.1.2 `--config`

The `--config` option determines which file in the **config/** directory contains the makefile environment for your build. E.g. to use the settings in **config/config.i686**, type

```
./configure --config=i686
```

If you give no `--config` option the default value is taken from the environment variable $MACHINE. If $MACHINE is not set the output of **uname -m** is used.

Note that some configuration files build Edge in single-precision, others in double precision, some compile Edge with debug flags, others compile with heavy optimization. Use the **config** file that best suits your needs.

If there is no suitable file in **config/** for your platform you may need to create a new config-file. Use one of the existing files in **config/** as template and save the file (e.g. as **config/config.custom**). Then configure with `--config=custom`. See the comments inside the config-files for more hints.

2.4.1.3 `--prefix`

The configure script also determines where Edge will be installed. The `--prefix` option determines this. E.g. to put the Edge binaries in **$HOME/bin** type

```
./configure --prefix=\$HOME --config=i686
```

If you give no `--prefix` option the default value is the build directory itself.

The prefix value may also be overridden later by specifying a prefix directory as an argument to "make install" (see 2.4.3).

2.4.1.4 `--with` and `--without`

The configure script searches certain standard directories to find optional external libraries. For more fine-grained control of where to find such libraries, use the `--with` and `--without` options. Type configure –help for more help on this subject.

2.4.1.5 `--program-suffix`

This option adds a suffix to all executable files. This may be convenient when installing several versions of Edge to the same location. E.g. to install a single-precision compile with program suffix `debug` type:

```
./configure --config=i686\_intel\_s --program-suffix=\_s
```

The command to run this installation of edge will then be **edge_run_s** instead of **edge_run**.

2.4.2 Compiling

Once Edge is configured, compiling is easy. Simply type

```
gmake
```

and wait for the build process to finish. You must use gmake (**GNU make**). Other make implementations are unlikely to work.

If you only want to build a single directory you can run gmake in that directory

```
cd solver/basic
gmake
```

To only build a single Fortran file **filename.f90** write

```
gmake filename.o
```

To remove all built object files and executable files, do

```
gmake clean
```

To rebuild dependency information write

```
gmake depend
```

This requires Python 2.

2.4.3 Installing

If the compilation finished without errors Edge can be installed.

```
gmake install
```

By default the binaries will be installed into the directory specified with the `--prefix` option to the **configure** script, but this value can be overridden by specifying a new prefix value directly to make:

```
gmake install prefix=/opt/local
```

To only install the contents of a subdirectory (e.g. solver) you may run

```
gmake install
```

from within that directory.

2.4.4 Build Java Based GUI (optional).

If you have the necessary prerequisites you may build Xedge, the Java GUI to Edge.

```
gmake gui.all
gmake gui.install
```

2.4.5 Testing your Installation

Your PATH environment variable must be set to the installation directory in order to run Edge. For convenience there is a special target `path` that prints any necessary PATH environment variable changes

```
gmake path
```

A simple first test to verify that Edge has been built ok is to run **ffalist** on any binary ffa file to view it's content e.g.

```
ffalist examples/rae_euler/mesh.bmsh
```

The output should be a listing with columns describing type, size and name of the datasets contained in this file.

If you wish, you can run Edge with the test cases found in the examples directory. See instructions for running Edge in Section 3.

3 Running Edge

3.1 Steps to Take to Run Edge

To start a computation with Edge all that is necessary is a grid file. The format for that grid file is described below. The steps to take to make a computation are the following:

1. Set up an input file with relevant parameters for your computation.
2. Set proper boundary conditions.
3. Run the preprocessor.
4. Run the flow solver.
5. Post processing.

Most files, input and output, are written in the FFA-format described in Section 3.3. The different files are explained in Section 3.4.

3.1.1 Setting Up an Input File

An input file with all global parameters is used to control the execution of all programs available in Edge. The input file contains necessary parameters and controls not only the flow solver but also the preprocessor and many of the helper programs. The description below relies on editing the input file with a text editor. The editing may also be done with the GUI Xedge if available.

An example input file with default setup can be found in **examples/default_edge.ainp** and the user should copy this file to his working directory and give it a relevant name and keep the suffix **.ainp**. The most relevant parameters that usually need to be modified are

1. File names assigned to the variables CFIMSH, CFIEDG, CFIOUT, CFIINI, CFIBOC, CFIRES.
2. Free stream values of the primitive variables PFREE, TFEE, UFREE, VFREE, WFREE.
3. Parameter for an Euler/Navier–Stokes calculation INSEUL.
4. For a viscous calculation the free stream viscosity RMU and specification for laminar/turbulent calculation ITURB.
5. Sequential or parallel computation by specifying NPART $= 1$.

Code	Meaning
	Connectivity boundary conditions
1	Periodic boundary
2	Rotation symmetry condition
3	Propeller disk boundary
	Wall boundary conditions
11	Euler wall
12	Adiabatic wall
13	Isothermal wall
	Symmetry boundary conditions
21	Symmetry
	External boundary conditions
31	Farfield
32	Farfield with vortex correction in 2D
33	Total states inlet (original type)
34	Total states inlet, pressure extrapolated
35	Incompressible inlet
36	Pressure outlet
37	Extrapolation
38	Free stream
39	Propulsion inlet
40	Propulsion outlet
41	Nacelle inlet
42	Nacelle exhaust
43	Mass flow inlet
44	Mass flow outlet
45	Mass flow outlet- time relaxation type
46	Mach number outlet

Table 3. Boundary conditions codes.

There is a large number of parameters (> 100) for different options in the preprocessor, numerical schemes in the flow solver, turbulence models, etc. More parameters than the above mentioned may have to be changed. A more complete list of available options for all programs is given in Appendix A. The values of the different parameters are given default values that should for most cases work well and need therefore not to be adjusted.

It is recommended to use different input files and different file names for computations with different conditions.

3.1.2 Specifying Boundary Conditions

The boundary conditions must be specified for each boundary that is given in the grid. To set the boundary conditions, execute

```
bound
```

and the program will ask for the input file and a specification of a boundary type for each boundary. The available boundary conditions are listed in Table 3.

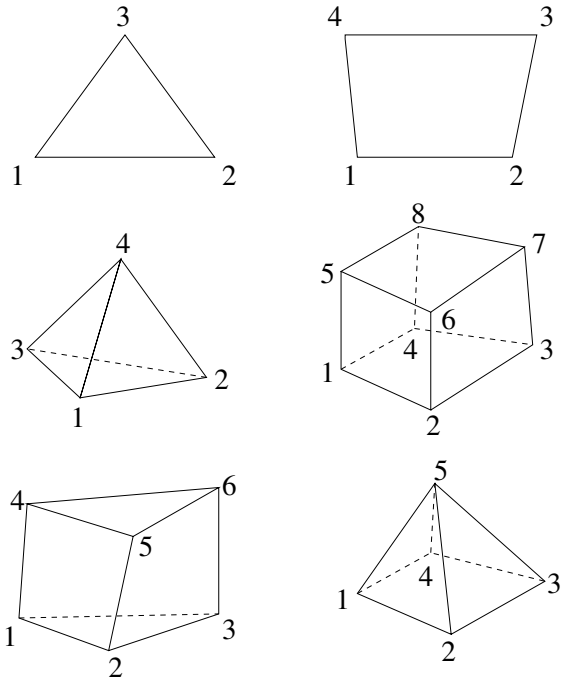


Figure 2. Enumeration of corner points for the volume elements: triangle, quadrilateral, tetrahedron, hexahedron, prismatic element and pyramid.

Once all boundary conditions are specified **bound** allows the user to change free stream values on the boundaries. It is only necessary to supply these values if they deviate from the free stream values given in the input file. If no values are supplied, the values from the input file are used.

The output from this program is written to a file with the name given by the variable CFIBOC.

3.1.3 Running the Preprocessor

The preprocessor is started with the command

```
preprocessor inputfile
```

In the first step of the preprocessor, the elements are discretized to control volumes surrounding each vertex. The volume elements accepted so far are triangle, quadrilateral, tetrahedron, hexahedron and prismatic element. The enumeration of these elements is depicted in Figure 2. The boundary elements are bar, triangle and quadrilateral. The orientation of boundary elements is automatically adjusted by the preprocessor to point outward from the computational domain.

In the second step, the preprocessor fuses control volumes to coarser grid cells, to be used for multigrid. The input parameter NLEVEL determines how many grid levels that are created.

If the input parameter NPART is greater than 1 then a partitioning of the grid is performed, enabling parallel computations on NPART processors.

The input parameter COLTYPE controls the reordering algorithm for nodes and edges. Reordering is done in order to improve performance of the solver by utilizing cache memory or vector computers better.

If vector coloring is selected the edgelist is permuted and divided into a number of colors in order to improve performance on vector machines. This is done in the **coloring** application which is called from the **preprocessor** script. Node order is not changed by this algorithm.

If the input parameter PDUAL is set to 1, **EnSight** files for each grid level with the extension **.dgrd** will be created. For 2D grids the dual grid will be plotted and for 3D grids the dual boundary grid will be plotted.

If cache reordering is selected both nodes and edges are reordered to improve performance on a modern CPU with cache memory. This is done in the **reorder** application which is called from the **preprocessor** script. Cache reordering is only done if also NPART is larger than 1. The reason is that there is no mechanism that restores the node order of the solution file for serial computations. If the node order was changed for a serial computation the node order of the solution file would not match the node order of the mesh file. For parallel computations **merge_partitions** restores the node order when merging the partitioned solution files into a single file.

The output filename is given by the variable CFIEDG. For a parallel computation (NPART $>$ 1) NPART files are output with extension **_p1**, ... **_pNPART** to the file name for each partitioning. Each of the processors in the flow solver will read a separate file and thus treat one partition per processor.

3.1.4 Running the Flow Solver

To run the solver, execute the solver script. For sequential execution, the name of the script is **edge_run**. For parallel execution (using MPI) the name is **edge_mpi_run**. Both scripts requires the input file as parameter, the parallel script also require the number of processors/partitions.

To control the execution of **Edge** one can store the file **update.ainp** in the running directory. The file should contain the UPDATE dataset defined as follows:

```
UPDATE,N,0,0,1
  IBREAK,I,1,1,0
  3
```

where the value of the parameter IBREAK determines what Edge will do. The following values of IBREAK apply

1. (Converged – for internal use only)
2. (Error / crashed – for internal use only)
3. Save and stop
4. Kill

In this manner any Edge parameter can be updated by adding the parameter in the UPDATE dataset with the new value.

When Edge has encountered the file, it will be removed.

3.1.5 Postprocessing

There is no postprocessing program contained in the Edge system. There are however three programs to export data to the commercial flow visualization tool **EnSight**. The program **ffa2engold** outputs the grid/solution on the EnSight Gold format and **uvol/ubody** output data on the EnSight 5 format. The former format is recommended by the EnSight developers. The EnSight Gold converter also has some additional features.

There is a program called **pres** to output the solution on a boundary in ASCII format on a data file.

The program **plotres** can be used to monitor the convergence and plots the residual and integrated forces and moment. It requires the open source software **Xmgr**.

3.1.6 Running Edge in Parallel

A parallel computation can be carried out by splitting the input grid into a desired number of partitions NPART > 1. The intention is that a parallel calculation shall produce the same output and be handled in the same way as a sequential calculation, *i*.e. a user should not notice any difference except for the specification of NPART > 1, choice of flow solver script (**edge_mpi_run** in stead of **edge_run**) and the computational speed up.

The **preprocessor** outputs a file for each partition/processor to be read by each of the flow solver processes. During the iterations, each of the processes outputs separate solution files that, at the end of the calculation, are merged into a single solution file. The resulting flow solution is therefore the same as the flow solution from a sequential computation.

3.1.7 Running the Grid Adaption

When a solution is obtained the grid can be adapted to the solution with the command

```
hadaption inputfile adapted_mesh.bmsh
```

The original grid is then h-refined to a new output grid (**adapted_mesh.bmsh**). The old cells are divided into smaller cells. In each adaption run the edges can only be divided once. A tetrahedra is divided into either two, four or eight new tetrahedron. An optional output file (**new_tetrahedra_mesh.bmsh**) contains all the tetrahedra created by the h-refinement, which is useful to check what the refinement has done.

The grid adaption is controlled by the following parameters in the input file:

- CELMAX – Maximum cell size (dimensional). Edges longer than this value will always be divided into two edges with a new node in the middle.
- CELMIN – Minimum cell size (dimensional). Edges shorter than this value will never be divided.
- CELFLO – Maximum indicator change in one cell. Normal value is 0.02–0.04. This means that the edges will be divided if the difference in the flow variables, normalized with the free stream values, between the two end nodes is larger than 2–4% (used only if SENSOR equals 0 or 1).

- RMAGNI – Maximum change number of nodes maximum by this factor. The grid adaption will create a grid with more nodes than the previous grid. This factor sets the maximum allowed number of nodes compared to the previous grid. If the adaption gives more nodes than this value no new grid will be created. The user can then either increase this value, or increase for example CELFLO in order to reduce the adaption sensitivity. If every tetrahedra are refined into 8 new ones RMAGNI must be larger than 8.0.

- IPROJ – If IPROJ equals 1, the nodes created on the boundaries are projected to an approximation of the original geometry. Otherwise, the nodes lie on the surface elements of the original grid.

- SENSOR – Indicates which method to use to select the cells to be refined:

  0 Same as 1, but with additional y^+ adaption for grids with prismatic layers. First layer is adapted to have y^+ equal to 1.0.

  1 Default method, the edges are divided in two equally pieces if the difference in the flow variables (ρ, u, v, w, p) between the two end nodes is larger than CELFLO.

  2 The edges are divided if their two end nodes have a total pressure ratio lower than THRESHOLD.

  3 The edges are divided if their two end nodes have an entropy production larger than THRESHOLD.

  4 An eigenvalue analysis of the velocity gradient tensor is performed at each node. The edges are divided if their two end nodes have a reduced velocity lower than THRESHOLD. This method is only relevant for flows with vortices and requires a postprocessing file (CFIPOST) that contains the gradients of the velocity. (See the **Edge Theoretical Formulation** manual for more details).

- THRESHOLD – Threshold value to start the refinement. Only used if SENSOR is larger than 1. Typical values are 0.950–0.999 for the total pressure ratio, 1–5 for the entropy production and 0.1–0.8 for the reduced velocity.

- RADIUS – Defines the shape of a box that limits where the refinement is performed. If RADIUS is positive, it is the radius of a cylinder. If RADIUS equals zero, the box is an hexahedra. Several RADIUS parameters can be given to define several refinement boxes.

- BOXCOOR – If RADIUS equals zero, BOXCOOR contains the coordinates of two points defining a diagonal of the refinement box. If RADIUS is positive, BOXCOOR contains the coordinates of the two end points of the cylinder axis. The format is X_{min} X_{max} Y_{min} Y_{max} $[Z_{\text{min}} Z_{\text{max}}]$. If $X_{\text{min}} = X_{\text{max}}$ and $Y_{\text{min}} = Y_{\text{max}}$ and $Z_{\text{min}} = Z_{\text{max}}$, no box is created.

- ADAPTCASE – H-refinement and/or y^+ adaption for SENSOR $= 0$.

  1 Only h-refinement.

  2 Only y^+ adaption of prismatic layer.

  3 Both h-refinement and y^+ adaption, this is the same as alternative 1 for a grid without a prismatic layer.

- YPLUS – Desired y^+ value for the first prismatic cell.

In order to run the flow solver with the adapted grid the preprocessor must first be executed on the new grid. Update the grid file name, CIFMSH, in the inputfile and then run the preprocessor.

3.1.8 Running Xedge

To start Xedge write

```
xedge
```

When working with Xedge the same steps are followed as when running Edge. The work flow goes from left to right in the GUI menus and the relevant options will get accessible as the necessary files are generated. A detailed description of Xedge is found in the HTML help pages accessible in Xedge, in the **Edge Graphical User Interface** manual or in Chevalier (2005).

3.2 List of all Programs in Edge

Below is a list with all the programs available in the current Edge distribution. All these programs are operated from the command line by entering the program name, with any required arguments. In many cases the program command has only one argument, the name of the main Edge input file.

To obtain help-text for any of these programs, simply enter the program name. For the more recent programs, the help-text gives complete instructions for running the program. For the aeroelastics programs, there is also an `ae-help` command, which prints a summary of the **ae** program set.

3.3 The FFA Dataset/Fileformat

With a few exceptions, all data files for Edge are FFA-format. The acronym "FFA" stands for "Flexible Format Architecture", which is a general datastandard defined by FFA, Sweden. The FFA-format defines a structure for hierarchical datasets of a recursive, linked list, form. These datasets can be represented in most programming languages and file systems.

The Fortran 90 representation of FFA-format is used in Edge and the FOI developed IR-ray tracing code **Sigge**. A Matlab representation, based on cell arrays, is used in the FFA Matlab Toolbox. Data files in FFA-format may be binary (IEEE bigendian) or represented in a plain ASCII format. One of the many advantages of the FFA-format is that binary files are generally portable.

The basic component in the FFA-format is the FFA *dataset*. A dataset is divided into an *identifier* set, comprising five *descriptors* and the *data* itself. A dataset can have one or

Name	Function
aedbelast	Transforms structural grid modeshapes to moving surface displacements.
aedbrot	Constructs moving surface displacements for general rigid body rotations.
aedis2tri	Transfers surface displacement data to a Tritet format used by aetribend.
ae-help	Print a summary of all the "ae series" programs.
aeputpert	Constructs a modal shape data file containing perturbation fields.
aetribend	Constructs a deformed mesh matching a given matching surface displacement.
aexbset	Extracts a movable surface set of nodes and boundaries.
bound	Interactive program to set the boundary conditions.
cgns2ffa	Converts grid/solution from CGNS format to FFA-format.
edge_mpi_run	Starts flow solver in parallel with mpi.
edge_run	Starts flow solver in single precision.
ffa23	Converts a 2D grid and/or solution to 3D by adding a parallel plane in z.
ffa2cgns	Converts grid/solution from FFA-format to CGNS format.
ffa2cgns_fluent	Converts an unstructured file from FFA-format to Fluent CGNS format.
ffa2engold	Exports grid and solution to the EnSight Gold format.
ffa2tau	Converts a 3D grid/solution to the tau (NetCDF) format.
ffa2tgrid	Converts a grid to the fluent tgrid format.
ffa32	Converts a 3D grid with two planes to a 2D grid.
ffaa2b	Converts any ASCII FFA-format file to binary.
ffab2a	Converts any binary FFA-format file to ASCII.
ffa_checkendian	Check binary file format whether it is on big or little endian.
ffad2r	Converts real data in double precision to single precision.
ffajdiff	Java tool to display difference between two FFA-format files.
ffajedit	Java tool to edit an FFA-format file.
ffalist	Lists the contents of any file in FFA-format.
ffamirror	Mirrors a grid file in its plane(s) of symmetry thus removed.
ffanode	Interactive program to print out the coordinates for a given node.
ffaucut	Cuts a boundary with a plane in a 3D problem.
ffauinterpol	Interpolates the solution based on the output from ffaucut, ffauline.
ffauline	Interactive program to extract data along a line.
ffauplane	Main program for cutting the edge mesh file (boundaries and volume).
ffaview	View and manipulate FFA-files using a GUI. Requires Python 2, pygtk and numarray packages.
ffaxyz	Interpolates an unstructured grid to user specified XYZ points.
fluent_cgns2ffa	Converts an unstructured file from Fluent CGNS to FFA-format
force	Computes forces and moments.
merge_boundaries	Merges and reduces number of boundaries in a mesh file.
merge_partitions	Merge solution files from a parallel run. This is called from edge_mpi_run.
meshdeform	Utility for mesh deformation.
plotmot	Plots modal timeseries tabulation data (can also be used from Matlab).
plotres	XMGR program to plot convergence history.
preprocessor	Produces dual grid, agglomeration and partitioning for parallel computations.
pres	Extracts data on a boundary column by column.
scale	Scales a given grid.
split_partitions	Split solution file into its partitions for parallel computations. Called from edge_mpi_run.
stout	Strips Edge standard-output data for plotting.
struct2unstruct	Converts a Euranus mesh file with boundary conditions to an Edge mesh file.
tau2ffa	Converts grid/solution from tau (NetCDF) to FFA-format.
tecgridbound	Exports the boundaries of a 3D grid to Tecplot format.
tecgridvol	Exports the interior and boundaries of a 3D grid to Tecplot format.
tecsolbound	Exports the boundaries of a 3D solution to Tecplot format.

Table 4. List of all available programs/scripts in Edge.

Name	Function
tecsolvol	Exports the interioir and boundaries of a 3D solution to Tecplot format.
transform_2_5d	Main program to convert a 2D .bedg file to compute in 2_5D.
ubody	Exports the boundaries of a 3D grid and solution to EnSight 5 format.
uvol	Exports the interior and boundaries of a grid and solution to EnSight 5 format.
VG_identify	Main program for specifying vortex generators (VG's) in Edge.

Table 4. Continued. List of all available programs/scripts in Edge.

several sub datasets, each of which is also an FFA dataset. In this way a hierarchical, tree like structure is assembled.

The five descriptors in each identifier set are

1. *name* (character*16): the name of the dataset
2. *type* (character*4): a four-character code as in Table 5
3. *ndim* (integer*4): the "dimension" of the data
4. *nsize* (integer*4): the "size" of the data
5. *nsub* (integer*4): the number of sub datasets

The data in each dataset is effectively a matrix, with *nsize* rows and *ndim* columns, the elements of which may be of numeric or character type. The four character *type* code denotes the type, function and location of the data. The fourth character is so far unused. Any dataset for which *nsize* or *ndim* is zero is a null dataset however a special type 'N' is also defined.

The source code module for the functions which handle FFA-format datasets within the Edge code is located in the directory **utils/**. Programmers who need to use the FFA-format in their own code, for example a format conversion routine, need only this code. A short programmers guide is provided in **doc/edge_util.html**.

Python programmers can use the module **ffa.py** located in the **bin/** directory. This module contains functionality to read, write and manipulate FFA datasets and files.

FFA-format data files can be inspected using the Edge helper program, **ffalist**, which prints a summary of the file data structure, including the first three elements of each data matrix. The program is the most frequently used in the whole Edge system and examples of output from the **ffalist** can be found throughout this manual. Two other useful programs are the ASCII - binary converters **ffaa2b** and **ffab2a**. Many more advanced functions for comparing and manipulating FFA-format data are available in the FFA Matlab Toolbox and in Xedge, the Edge GUI. Finally there is a python program **ffaview.py** which can also view and manipulate FFA files through a GUI (requires Python 2 with the **pygtk** package installed).

FFA file naming convention dictates that FFA binary files have names ending in **.b???** and FFA ASCII files have names ending in **.a???**, where "**?**" denotes any single character. Examples of this convention can be seen in Table1 on page 14. This filename convention is strictly enforced.

FFA-format binary files are portable between Unix/Linux machines with IEEE bigendian format. To achieve this, the whole Edge system must be compiled using the appropriate compiler settings defined using the files in the **config/** directory. An FFA dataset is written to a binary file using the following Fortran statements

Code	Meaning
B	binary*1
I	integer*4
R	real*4
D	real*8
C	complex(4+4)
Z	double complex (8+8)
A	character*1
S	character*16
L	character*72
N	No data

Table 5. FFA-format: type(1:1) data “type”.

Code	Meaning
I	Descriptor
F	Field
B	Boundary information
C	Constants
T	Data tables
L	Data list

Table 6. FFA-format: type(2:2) data “function”.

```
WRITE(UNIT)CNAME,CTYPE,NDIM,NSIZE,NSUB
WRITE(UNIT)(DATA(JSIZE,JDIM),JSIZE=1,NSIZE),JDIM=1,NDIM)
```

Each dataset is written as two Fortran records, a descriptor record, followed by a data record. The detailed, byte level, structure of these records is visible in the code for programs **ffa_forti.m** and **ffa_forto.m** in the FFA Matlab Toolbox. Attempts to use Edge on Windows platforms may at present result in incompatible Fortran record structures.

Occasionally one may encounter non-standard *smallendian* “FFA-format” files, usually due to an incorrect compilation. The endian status of any FFA-format file can be checked or switched using the programs **ffa_checkendian** and **ffa_swapendian** which are supplied with Edge.

FFA-format ASCII files are used mainly for input to Edge and other programs. One important example is the default main input file for Edge in **examples/** directory. The data structure and ordering of these files are exactly the same as the binary equivalents, except that they may contain comment lines. A comment line always starts with an asterisk (*) in the first column. Comment lines are ignored by the Edge subroutine that reads FFA data.

The task of creating and editing FFA-format ASCII files can be made easier by the use of syntax highlighting (colored text). Syntax highlighting definitions for the **Nedit** editor are available. For creating and editing the main Edge input file, the use of the Edge GUI is recommended.

Code	Meaning
N	Node values
F	Face values
C	Cell centre values

Table 7. FFA-format: type(3:3) data "location".

Files required	Application	Files generated
Edge.ainp Edge.bmsh	**bound**	Edge.aboc
Edge.ainp Edge.bmsh Edge.aboc	**preprocessor**	Edge.bedg(_p?)
Edge.ainp Edge.bedg Edge.aboc Edge.bini	**edge_***	Edge.bout Post.bout Edge.bres Edge.log
Additional files for modal Aeroelastic applications		
Edge.bmos Edge.amot Edge.amop	**edge_***	Edge_timestep.bout Edge.bres (modal data)
Additional files for adjoint flow calculations		
Edge_flow.bout	**edge_***	Edge_adjoint.bout

Table 8. Required/generated files for Edge applications.

3.4 Files Generated/Required by Edge

To start Edge, the only essential input data is a grid file (**.bmsh**). A number of example grid files can be found in the directory **examples/**. In Table 8 the required input files and generated files are listed for Edge and the two main supporting programs. As far as possible, the file names are as given in the default input file **$(Edge_HOME)/lib/default_edge.ainp**.

The main files listed table 8 in are explained below. The additional files for aeroelastic applications are described in Section 3.8. All these files are FFA-format files, as described in Section 3.3. The file names in the main Edge input file can be set to any value, provided that the same four character extension (like **.bmos**) is used. The main Edge input file is **Edge.ainp**.

The main Edge input file is most easily set up using the Edge graphical user interface, which is started using the command `xedge`, however, it can also be prepared by direct manual editing. The file is an FFA-format ASCII file (suffix **.ainp**) and should initially be a copy of the default input file **examples/default_edge.ainp** in which all input variables are set to default values. The input file is constructed by manually resetting the appropriate parameters. Typically, an existing case-specific input file is used, together with the default file as a template and the editing process is assisted with a graphical difference tool, such as **tkdiff** for example.

A complete list and explanations of all main-input variables, is given in Appendix A. Many of these variables are also described in the **Edge Theoretical Formulation** manual.

```
N       0 x 0   9/  unstr_grid_data
L       1 x 1          title  = "12.2_adapt_1"
RF  88703 x 2          coordinates  =  1.14088404 1.140531
N       0 x 0   4/     region
L       1 x 1             region_name  = "volume"
N       0 x 0   2/        element_group
L       1 x 1                element_type  = "tria3"
IF  87235 x 3                element_nodes  =  36736 37929
N       0 x 0   2/        element_group
L       1 x 1                element_type  = "quad4"
IF  44457 x 4                element_nodes  =  1 2 3
N       0 x 0   3/     boundary
L       1 x 1             boundary_name  = "slat:p"
N       0 x 0   2/        belem_group
L       1 x 1                bound_elem_type  = "bar2"
IF    371 x 2                bound_elem_nodes  =  279 280
N       0 x 0   3/     boundary
L       1 x 1             boundary_name  = "wing:p"
N       0 x 0   2/        belem_group
L       1 x 1                bound_elem_type  = "bar2"
IF    591 x 2                bound_elem_nodes  =  650 651
N       0 x 0   3/     boundary
L       1 x 1             boundary_name  = "flap:p"
N       0 x 0   2/        belem_group
L       1 x 1                bound_elem_type  = "bar2"
IF    278 x 2                bound_elem_nodes  =  1 2 3
N       0 x 0   3/     boundary
L       1 x 1             boundary_name  = "outer"
N       0 x 0   2/        belem_group
L       1 x 1                bound_elem_type  = "bar2"
IF     21 x 2                bound_elem_nodes  =  46104
...
```

Figure 3. A typical grid (**.bmsh**) file.

3.4.1 The Input Grid File **Edge.bmsh**

A grid file is required to start any application of Edge and must be prepared using a mesh generator, such as the FFA-developed system, **Tritet**, or commercially available tools such as **ICEM**. The grid file is written in binary FFA-format (suffix **.bmsh**). A typical file listing is given in Figure 3. Illustration 3 is printed using the program **ffalist** and contains the type of the dataset to the left, the size and dimension in the middle and the and either the number of subsets or the first data to the right. The grid file contains a list of coordinates, a list for each element type (triangles tria3 and quadrilaterals quad4) with node numbers referring to the list of coordinates. Each of the boundaries, four in the example, should be given a name (boundary_name) and a type (bound_elem_type) and list of nodes (bound_elem_nodes) for each type.

3.4.2 The Boundary Condition File **Edge.aboc**

This file is generated by the program bound and contains, for each boundary, a complete description of the boundary condition to be used. This is a small file and is therefore usually an ASCII file. An example of it is given in Figure 4.

```
N         0 x 0   4/  boundary_data
NB        0 x 0   3/     boundary
L         1 x 1             b_name  = "slat:p"
L         1 x 1             b_class  = "wall"
L         1 x 1             b_type  = "weak adiabatic"
NB        0 x 0   3/     boundary
L         1 x 1             b_name  = "wing:p"
L         1 x 1             b_class  = "wall"
L         1 x 1             b_type  = "weak adiabatic"
NB        0 x 0   3/     boundary
L         1 x 1             b_name  = "flap:p"
L         1 x 1             b_class  = "wall"
L         1 x 1             b_type  = "weak adiabatic"
NB        0 x 0   3/     boundary
L         1 x 1             b_name  = "outer"
L         1 x 1             b_class  = "external"
L         1 x 1             b_type  = "weak characteristic"
...
```

Figure 4. A typical boundary condition (**.aboc**) file.

3.4.3 The Edge File **Edge.bedg**

The edge file is produced by the preprocessor. The element information from the grid file has been processed and complemented or replaced with edge and node information. For each edge the two nodes are given (edge_nodes) as well as the control surface (edge_surfaces) building up the control volumes. For each boundary the boundary nodes (b_nodes), the most orthogonal inner node (b_inner_nodes) and a control surface is computed. The preprocessor also agglomerates the control volumes to coarser grids so the same type of information (node_f2c) is repeated for each grid level (grid) together with connectivity information between the grids.

In case when a parallel computation is to be carried out (NPART > 1) then NPART files are produced for each partition/processor with names **Edge.bedg_p1**,. . . ,**Edge_pNPART**.

This file is usually given in binary due to its size being larger than the grid file, an example file is listed in Figure 5. Due to the amount of information the output is truncated and only the fine grid information is given.

3.4.4 The Solution File **Edge.bout** (**Edge.bini)**

The file **Edge.bout** is generated from the flow solver and contains the nodal values of all primitive variables in the flow solution. The nodal ordering is the same as in the (**.bmsh**) grid file. This file may be used to *restart* an Edge computation from a the saved solution (INPRES $= 2$). This file may also be used as an initialization solution file, suffix **.bini** (INPRES $= 1$). The contents of the file depend on the physical model used: Euler/Navier–Stokes, laminar/turbulent, steady/unsteady etc.

A listing of an example output from a turbulent steady state solution is given in Figure 6.

For time accurate solutions, there are *two* groups of flow field datasets, corresponding to the current and previous time steps, and these are located under “time” datasets. For a *restart*, both of these time levels are required, together with the (.bres) time history file. If used for an *initialization*, as a **.bini** file, only the first “time” group is used and it is treated as steady state data.

```
N         0 x 0   3/  dual_grid_data
L         1 x 1          title  = "12.2_adapt_1"
I         1 x 1          n_dim  =  2
NB        0 x 0  14/     grid
IF   220397 x 2             edge_nodes  =  1 1 2
DF    88703 x 1             volumes  =  2.4808954890E-10
I         1 x 1             n_nodes  =  88703
DF    88703 x 2             coordinates  =  1.1408840417861
I         1 x 1             n_edges  =  220397
DF   220397 x 2             edge_surfaces  =  -8.195638E-08
I         1 x 1             n_edge_colors  =  2
IF        2 x 2             color_indices  =  1 1262
I         1 x 1             n_b_colors  =  1
I         1 x 1             n_bound  =  4
NB        0 x 0   5/        boundary
LB        1 x 1                b_name  = "slat:p"
I         1 x 1                n_b_nodes  =  371
IF      371 x 1                b_nodes  =  279 280 281
DF      371 x 2                b_surfaces  =  -1.011956E-04
IF      371 x 1                b_inner_nodes  =  1519 1520
NB        0 x 0   5/        boundary
LB        1 x 1                b_name  = "wing:p"
I         1 x 1                n_b_nodes  =  591
IF      591 x 1                b_nodes  =  650 651 652
DF      591 x 2                b_surfaces  =  -1.328219E-04
IF      591 x 1                b_inner_nodes  =  1890 1891
NB        0 x 0   5/        boundary
LB        1 x 1                b_name  = "flap:p"
I         1 x 1                n_b_nodes  =  278
IF      278 x 1                b_nodes  =  1 2 3
DF      278 x 2                b_surfaces  =  -6.411895E-04
IF      278 x 1                b_inner_nodes  =  1241 1242
NB        0 x 0   5/        boundary
LB        1 x 1                b_name  = "outer"
I         1 x 1                n_b_nodes  =  21
IF       21 x 1                b_nodes  =  46090 46091
DF       21 x 2                b_surfaces  =  28.4852609634
IF       21 x 1                b_inner_nodes  =  88690 88700
...
```

Figure 5. A typical edge (**.bedg**) file.

```
N            0 x 0       8/  solution
L            1 x 1              brand  = "Edge 4.0.0 www.edge.foi.se"
LI           1 x 1              title  = "Edge, an edge-based flow-solver from FFA."
LI           1 x 1              date  = " 070204 - 23:19:41"
LI           1 x 1              author  = ""
LI           1 x 1              program  = "Edge"
LI           1 x 1              grid_ref  = "Grid collected from ffanet-files"
N            0 x 0       8/     free_stream_data
DC           1 x 1                 pressure  =    37643.8007812500
DC           1 x 1                 temperature  =    300.000000000000
DC           1 x 3                 velocity  =    261.516998291016
DC           1 x 1                 gamma  =    1.39999997615814
DC           1 x 1                 rgas  =    286.999987781048
DC           1 x 1                 mu_reference  =   1.845999940996990E-005
DC           1 x 1                 prantl_number  =   0.720000028610229
DC           1 x 1                 dynamic_pressure  =    14980.7743211015
N            0 x 0       6/     block
DF       22088 x 1                 density  =   0.434306166672034
DF       22088 x 3                 velocity  =    255.959142902464
DF       22088 x 1                 pressure  =    37609.3537068229
DF       22088 x 1                 turb_kin_energy  =    17.8736499360706
DF       22088 x 1                 turb_omega  =    314.440368967509
DF       22088 x 1                 turb_viscosity  =   4.194888755189772E-002
...
```

Figure 6. A typical solution (**.bout**) file.

3.4.5 The Sample Solution Files **Edge_timestp.bout**

The user has an option to write a "sample" output file similar to the **Edge.bout** file at specified time steps. The input variable IMULOU controls this output so that Edge writes an **Edge_timestep.bout** every IMULOU'th step. The file contains a time stamp. Figure 7 shows a typical intermediate solution file.

3.4.6 The Solution File for Post Processing **Post.bout**

The file **Post.bout** is generated whenever additional data on a separate file is desired (IPOST $>$ 0). The format is exactly the same as for the solution file **Edge.bout**, the contents is determined from the variable IPOST.

Several options for additional output of field data to the **Post.bout** exist, e.g. surface skin friction vector, surface heat transfer, wall distance field, laminar/turbulent field, gradients, residuals etc. Note that surface quantities like skin friction and heat transfer have values on wall boundaries only and are zero elsewhere. These quantities are scaled with dynamic pressure and made non-dimensional.

3.4.7 The Residual File **Edge.bres**

The residual file Edge.bres is output from the flow solver, it is written together with the flow solution file **Edge.bout**. This file is intended to monitor the rate of convergence (e.g. with the program **plotres**) and contents of this file contains residuals, integrated forces and moments plus a few additional quantities. An example file is given in Figure 8.

For time accurate solutions and aeroelastic applications, the file also contains time series for modal forces, modal coordinates and time information.

```
N          0 x 0   8/  solution
L          1 x 1          brand  = "Edge 3.2 www.edge.foi."
LI         1 x 1          title  = "Edge,Re=9m, alpha=12.2"
LI         1 x 1          date  = " 040506 - 11:14:39"
LI         1 x 1          author  = ""
LI         1 x 1          program  = "Edge"
LI         1 x 1          grid_ref  = ""
N          0 x 0   7/     free_stream_data
RC         1 x 1             pressure  =  100000.000
RC         1 x 1             temperature  =  300.000000
RC         1 x 3             velocity  =  67.8771973
RC         1 x 1             gamma  =  1.39999998
RC         1 x 1             rgas  =  287.000000
RC         1 x 1             mu_reference  =  8.9608638E-06
RC         1 x 1             prantl_number  =  0.720000029
N          0 x 0   6/     block
RF     88703 x 1             density  =  1.14204168
RF     88703 x 3             velocity  =  0.00000000E+00
RF     88703 x 1             pressure  =  99098.6484
RF     88703 x 1             turb_kin_energy  =  0.000E+00
RF     88703 x 1             turb_omega  =  3.79523712E+09
RF     88703 x 1             turb_viscosity  =  0.000E+00
...
```

Figure 7. A typical intermediate solution (**.bout**) file.

```
N          0 x 0  22/  time_history
IT       764 x 1          iteration_number  =  0 10 20
DT       764 x 1          Cl  =  2.715464574425870E-02
DT       764 x 1          Cd  =  0.5877895227793520 4.42009
DT       764 x 1          Cm  =  6.286313173730319E-02
DT       764 x 1          cpu_time  =  1.63099999999394
DT       764 x 1          work_units  =  0.00000000000E+00
DT       764 x 1          max_heattransfer  =  3.869822E-09
DT       764 x 1          max_temperature  =  299.9999999
DT       764 x 1          rho_rmsresidual  =  6.683030961
DT       764 x 1          rho_maxresidual  =  8.2024096726
DT       764 x 1          rhou_rmsresidual  =  8.497851336
DT       764 x 1          rhou_maxresidual  =  10.21657562
DT       764 x 1          rhov_rmsresidual  =  7.837199928
DT       764 x 1          rhov_maxresidual  =  9.571859964
DT       764 x 1          rhow_rmsresidual  =  -20.00000000
DT       764 x 1          rhow_maxresidual  =  -20.00000000
DT       764 x 1          rhoe_rmsresidual  =  12.16430666
DT       764 x 1          rhoe_maxresidual  =  13.67740256
DT       764 x 1          rhoq_01_rmsresid  =  6.373938039
DT       764 x 1          rhoq_01_maxresid  =  8.087178792
DT       764 x 1          rhoq_02_rmsresid  =  16.08406770
DT       764 x 1          rhoq_02_maxresid  =  18.28947518
...
```

Figure 8. A typical residual (**.bres**) file.

```
N        0 x 0   8/  solution
 L        1 x 1          brand  = "Edge 4.0.0 www.edge.foi.se"
 LI       1 x 1          title  = "Edge, an edge-based flow-solver. "
 LI       1 x 1          date   = " 060713 - 18:24:17"
 LI       1 x 1          author  = ""
 LI       1 x 1          program  = "Edge"
 LI       1 x 1          grid_ref  = "Grid converted from CGNS"
 N        0 x 0   8/     free_stream_data
 DC       1 x 1             pressure  =    100000.000000000
 DC       1 x 1             temperature  =    262.938300000000
 DC       1 x 3             velocity  =    272.478929818100
 DC       1 x 1             gamma  =    1.39999997615814
 DC       1 x 1             rgas  =    286.999987781048
 DC       1 x 1             mu_reference  =   1.993280000000000E-005
 DC       1 x 1             prantl_number  =   0.720000028610229
 DC       1 x 1             dynamic_pressure  =    49333.2196003349
 N        0 x 0   4/     block
 DF  226515 x 1             density  =   1.491653856239447E-004
 DF  226515 x 3             velocity  =  -8.664490392305818E-007
 DF  226515 x 1             pressure  =   8.148238739742469E-010
 DF  226515 x 3             grad  =  -6.890308927563643E-004
```

Figure 9. A typical solution file from an adjoint calculation (**.bout**) including nodal gradients.

3.4.8 The Sample Solution File **Edge_adjoint.bout**

The result from an adjoint flow calculation is saved in a usual **Edge_adjoint.bout** file where the density, velocity and pressure are the adjoint variables instead of the flow primitive variables. They are in fact the co-state variables of the flow equations expressed in conservative variables. The adjoint variables may be used for further calculating the shape gradient, for example, of the functional J associated with the adjoint problem (the type of the functional J is given via the input variable FUNTYP in the **.ainp** file dedicated to this calculation). If the ADJOPT variable is set to the value 2 (in the file **.ainp**), a post-processing of the adjoint variables delivers an additional field called gradient_set in the result file (such as in Figure 9). This is the gradient of the functional J with respect to the nodal coordinates of the mesh. It has therefore the size and dimension of the field coordinates as found in the mesh file **.bmsh** or the edge file **.bedg** associated to this problem. This result is usually used for calculating gradients of functionals with respect to design paramaters. In that case a mesh deformation program is required in order to map the displacements of the nodes in the entire domain of calculation from the displacements of the nodes on the shape only. It is also usual to parameterize the nodes on the shape to optimize instead of using the coordinates themselves as design parameters.

Suppose that an explicit linear mapping is used to deform the mesh for given coordinates at the boundary being optimized. We may denote it $X = Ay$, where X stands for the vector of nodal coordinates in the entire flow domain, y stands for the vector of nodal coordinates of the shape that is optimized, and A is the matrix of real coefficients associated with the linear mapping. For a given 'shape' (y) the inviscid flow (Euler equations) is solved giving the state variables w. The solution of the adjoint equations associated with the Euler equations, the flow state w, the shape y and the functional J (FUNTYP) is denoted w^*. Having set the ADJOPT variable to 2 in the **.ainp** file for the adjoint calculation tells Edge to compute the gradient of J with respect to X, which is done without computational effort once the adjoint problem has been solved (w^*). As named above, the gradient of J is saved in the result file **Edge.bout** under grad, we call it here ∇J_X. The gradient with respect to the shape (that is with respect to the vector y of coordinates) can then be simply obtained by the chain rule as

$\nabla J_y = A^T \nabla J_X.$

3.5 Hints and Recommendations

3.5.1 Specifying the Free Stream Viscosity and Reynolds Number

The free stream viscosity should be specified such that the desired Reynolds number is obtained for a viscous calculation (INSEUL $= 1$). This can be done two ways. If the free stream temperature, pressure (and hence density) and velocity are specified such that the desired free stream Mach number is obtained, the free stream viscosity μ_∞, (RMU) is computed from (1). Note that L represents a length in the *dimension of the grid* on which the Reynolds number is based.

$$Re = \frac{|U_\infty| L \rho_\infty}{\mu_\infty}. \tag{1}$$

Alternatively the free stream viscosity can be computed from the Sutherland law, see (2), commonly used for situations where no Reynolds number is given, e.g. internal flows.

$$\mu_\infty = S_1 \frac{T_\infty^{3/2}}{T_\infty + S_2}, \quad S_1 = 1.458 \cdot 10^{-6}, \quad S_2 = 110 \text{ K}. \tag{2}$$

The viscosity can vary with the temperature (ICOVIC $= 0$) as in (3),

$$\mu = \mu_\infty \left(\frac{T}{T_\infty} \right)^{3/2} \frac{T_\infty + 110}{T + 110}. \tag{3}$$

or it can be constant (ICOVIC $= 1$).

3.5.2 Turbulent Flow Calculations

There are three main types of turbulent calculations that can be performed by Edge: RANS, DES and other hybrid RANS-LES methods, as well as LES. The parameter ITURB in the input parameter file determines which model is used.

The DES, hybrid RANS-LES and LES modeling methods are unsteady approaches by nature. When used, *i*.e. setting ITURB $= 3$ or 4, the option for time-accurate computation must be switched on by setting the parameter ITIMAQ > 0 (ITIMAQ $= 1$ is recommended) and consequently other related parameters should be specified to enable a time-dependent calculation.

All RANS models are integrated down to the wall (low-Re approach) and the wall-normal grid resolution for the closest near-wall nodes must be of the order of $y^+ = 1$. With DES and other hybrid RANS-LES models, the wall-normal grid resolution is similar to the low-Re RANS approaches. Note that DES is a hybrid approach combining near-wall RANS modeling and off-wall LES modeling. With the Spalart-Allmaras DES model, it is preferable to have the near-wall RANS switched to LES at a location around the outer edge of the boundary layer.

3.5.2.1 RANS

RANS (Reynolds Averaged Navier–Stokes equations) models can be chosen by setting ITURB $= 2$. There are several RANS models to choose from. Each of these models has a number of parameters. It is recommended that these parameters are not changed by the user. These parameters are therefore not listed in the Appendix.

The following four models are recommended for RANS computations:

- Spalart-Allmaras one-eq model' Spalart & Allmaras (1994): One of the most widely used models for aeronautical flows. With only one turbulence transport equation, it is relatively simple and robust and gives reasonable results in attached and slightly separated flows.
- 'Menter SST k-omega' Menter (1994): One of the most widely used models. Somewhat more sophisticated than the Spalart Allmaras model, and also more general. It should give reasonable results for attached and separated flows.
- 'W&J EARSM + std k-omega' Wallin & Johansson (2000): A model that is able to capture more of the complex physics associated with rotation, curvature, three-dimensionality, non-equilibrium flow and separation. Reasonable robust and reliable in most flows. The default model in previous versions of Edge.
- 'W&J EARSM + Hellsten k-omega' Hellsten (1998): Comparable with the Wallin (2000) model, but the omega equation is redefined and recalibrated fully consistently with the EARSM assumption. This model is, thus, better behaved and slightly more accurate. The new default model.

In addition, there are two low-Reynolds number (LRN) k–ω models included in Edge, 'Wilcox LRN k-omega' and 'PDH LRN k-omega', which may be good alternatives for computing internal turbulent flows with moderate separation and/or reattachment.

The other models listed in Appendix A.3 may be tried if the above models fail to converge. A typical comparison of several RANS models for in Edge for a computation on an aircraft configuration can be found in Peng & Eliasson (2004).

Performing several computations with different turbulence models is good practice in order to assess the possible sensitivity to the modeling. Large differences between models indicates that the flow is highly critical with respect to the turbulence modeling, and that the results are unreliable. In this case, a comparison with experimental data may be necessary in order to determine which model is most accurate.

3.5.2.2 DES and Hybrid RANS-LES

The DES (Detached Eddy Simulation) or hybrid RANS-LES model is chosen by setting ITURB $= 3$. In this version, besides the 'Spalart-Allmaras DES' model, a hybrid RANS-LES model, 'Peng hybrid HYB0 model' is also available. The DES and the hybrid RANS-LES model has been developed for unsteady aerodynamic flows, characterized usually by massive boundary layer separation and/or after-body vortex shedding.

In applications of CFD analysis of flow control, aeroacoustics and fluid-structure interaction dealing with unsteady (time-dependent) flow phenomena, the DES and hybrid RANS-LES methods should be taken into consideration in the selection of modelling approaches. These

methods are able to give rich resolved flow structures and time-dependent turbulent fluctuations.

3.5.2.3 LES

LES (Large Eddy Simulation) is chosen by setting ITURB $= 4$. The model employed in LES is often termed "SubGrid Scale (SGS)" model. Three SGS models have been implemented in Edge with the 'Smagorinsky SGS model' Smagorinsky (1963) as the default. The other LES model is the SGS variant of the Spalart-Allmaras model, which has been implemented for testing the model constant, C_{des}, in the context of EDGE. When the grid is sufficiently fine, the S-A SGS model should perform similarly to the Smagorinsky SGS model. Another SGS model is the SGS k-equation model by Yoshizawa (1986). The grid resolution requirement in LES is very high in all three directions, particularly in the wall boundary layer, so that it may become infeasible for computations of typical aerodynamic configurations at high Reynolds numbers. When a computationally affordable grid can be applied for fundamental flows or for flows at moderate Reynolds number, LES may provide with much richer information and usually more accurate modelling than RANS approaches. Note that LES should be performed in the context of time-dependent and three-dimensional calculations.

It is recommended that LES is only used by experienced users or in consultancy with FOI.

3.5.2.4 Specification of Related Parameters

For external aerodynamic flow simulations, the free stream turbulence intensity, TUFREE, has a default value of TUFREE $= 0.001$ for RANS computations. For DES and LES, a value of about 10% of this value should be used. The parameter VRFREEis the ratio of the free stream eddy viscosity and the molecular viscosity. A value of VRFREE $= 1.0$ is taken as the default value in RANS. With the Spalart-Allmaras RANS model, as well as with DES and LES, a value of VRFREE $= 0.1$ is recommended.

For DES, hybrid RANS-LES and LES computations, the second-order central scheme for the spatial discretization of the momentum equations must be used, namely, IUPWIN $= 0$. For the turbulence transport equation, both schemes, second-order central or upwind schemes, can be selected by setting IUPWKZ $= 0$ or IUPWKZ $= 1$ correspondingly, but the second-order central scheme is recommended for DES, hybrid RANS-LES and LES computations. With the central scheme in DES and LES, the two parameters that control the artificial dissipation, VIS2 and VIS4 for the momentum equations and VIS2KZ and VIS4KZ for turbulence transport equation(s), may be specified with a value slightly smaller than that in RANS. A typical value of between 0.01–0.02 can be taken for VIS4 and VIS4KZ, respectively.

For low speed (low Mach number) flow calculations it is recommended to have a negative value of the variable TURFIX for an upwind discretization of the turbulent equations, TURFIX $= -1.1$ is recommended. A positive value may add too much dissipation and hence destroy the accuracy.

3.5.2.5 Specification of Transition

It is possible to specify explicitly the position at which the transition from laminar to turbulent flow takes place. The user should then specify the variable NLAMREG to the number

of laminar regions in the computational domain. A laminar region is a region of a wall boundary that can be specified by a number of limiting planes (LAMINAR_REGION) defined by a coordinate and a direction into the laminar region for each plane. The laminar region is then defined by the convex region built up by the planes.

It is possible to limit the laminar region close to the wall by the parameter MAXLAMDIST so that turbulent flow is obtained at nodes at larger distances.

Specification of laminar regions is error prone and it is therefore recommended that the user uses the post processing option IPOST $= 4$ in the flow solver to output the array specifying laminar/turbulent flow (0/1 respectively) as well as the distances to the walls.

3.5.3 Convergence Problems

The default setting for the numerical parameters usually provides good convergence rates and stable solutions. It may happen though, that the convergence is poor or there might even be cases that diverge.

There may be many reasons behind convergence problems. One of the first tests to find the reason behind it is to lower the CFL numbers (CFL, CFLVIS, CFLREF) to see if this stabilizes and improves the convergence. Another parameter may be to reduce the number of multigrid levels (NGRID) to see if the coarser grids has an overall influence.

It may be that the numerical dissipation is too small so the parameters VIS4, VIS2, VIS0, VIS4KZ, VIS2KZ should be increased. For upwind discretization the parameters LIMITE, ENTRFX, TURFIX are concerned.

In viscous calculations the coarsening ratio, CRATIOB, in the boundary layer can occasionally lead to instabilities. This quantity may then be reduced to CRATIOB $= 2$. As an additional consequence of this the coarser grids become finer and the overall computing time increases.

The residual smoothing has occasionally caused problems and it may be worthwhile to alter the value of RSMPAR to see the effect. This is rare though.

Another reason behind poor convergence is that the flow is actually unsteady in nature and hence there is no steady state solution. In this cases the unsteady option (ITIMAQ $= 1$) may be considered.

3.5.4 Time Accurate Simulation and Inner Iterations

Setting ITIMAQ $= 1$ enables time accurate simulation with implicit time stepping. There will be ITMAX time steps with global time step DELTAT. The driver of the implicit time stepping is a sequence of inner iterations by means of the dual time stepping method. The number of such inner steps, is controlled by three parameters in the input file: RESTAQ (see above) which is a convergence criterion on the residuals, ITMXAQ which is the maximum number of inner iterations to be used, and ITMNAQ which is the minimum number of such inner steps.

Usually ITMXAQ is the important parameter. This has to be large enough to ensure sufficient convergence of the flow variables at each physical time step before going on to the next. There is no general perception of what this number should be, although 30–100 inner iterations could be required for a full viscous computation. To approximately estimate the number of sub-iteration, one can observe the residual reduction at each physical time

step, which should carry on in an approximately cyclic manner when the solution moves from one time step to the next. Furthermore, with increasing sub-iteration number at each time step, the variation in the output parameters (e.g. the integrated lift and drag) should be negligibly small. DELTAT also influences the number of inner iterations ITMXAQ. In general, reducing the value of DELTAT (a smaller time step) may result in a reduction of the inner sub-iteration number, ITMXAQ. In DES and LES, the time resolution is important, for which an appropriate specification of the DELTAT value may affect both the numerical solution procedure and the resolution of the resolved turbulent structure. Therefore, caution must be taken in DES and LES with the value of DELTAT, which in some occasions may have to taken a smaller value than in RANS calculation for the same problem in order to get accurate physical resolution of the flow evolution.

It is also possible to make time accurate calculations with explicit Runge–Kutta time marching by imposing ITIMAQ $= 2$. The Runge–Kutta coefficients IRKCO have to be adapted so that the desired temporal accuracy is obtained, the default coefficients are optimal fr steady state calculations and are provide only first order accuracy. Explicit time marching requires fairly equal sizes of the grid elements since the maximum allowable time step is determined by the smallest control volume. Explicit time marching can not be combined with multigrid and implicit residual smoothing.

The physical time scales of the problem usually give a hint on how large DELTAT should be. For instance, there should be at least some 30–60 time steps per cycle of a harmonically oscillating component to be resolved.

Best practice is to plot all available global quantities, like aerodynamic forces (or generalized forces), to verify that they do converge within the inner iterations. In this respect moments are sharper probes than lift components.

Incorporated into the time accurate simulation (with unsteady RANS, DES or LES), a parameter, ITMEAN, is used to switch on ($= 1$) the time-averaging. For DES and LES, this parameter should be set as ITMEAN $= 1$, since the default output of the solution is for the instantaneous, resolved large-scale field. Setting ITMEAN $= 1$ enables to make time-averaging of the instantaneous field and gives output of the time-averaged mean field. Another parameter, ITERM, specifies the time step, at which the time-averaging process starts. This should be justified by looking into the time history of local solution variables in the solution procedure. Up to the specified time step at ITERM, the flow should have been fully developed. This can be observed by recording the time history of the flow variables at some typical locations. A proper value for ITERM is justified by the timestep at and after which the flow variable recoded with time at a typical location fluctuates around a constant mean value.

The parameters, NPTHIS, specifies the number of the locations where the time-history of the flow variables is recoded. The coordinates of the locations (x, y, z) are specified in the sub-dataset POINT_COOD, and another sub-dataset, POINT_INDEX, is used to specify the type of the recoded points: 'bound_point' indicates a point on the boundary, 'inter_point' for an interior point.

Name	Function
Periodic	A connectivity through a translational vector.
Rotation	A connectivity through a rotational matrix.
Propeller	Propeller disk model connectivity.

Table 9. Connectivity boundary conditions.

3.6 Boundary Conditions in Edge

The boundaries in the computational domain have been determined already when the grid was generated and can be found in the .bmsh file, Section 3.4.1 on page 29. The program bound creates an additional file with more detailed data for each boundary containing type of boundary condition, local free stream boundary values etc. This file has the suffix **.aboc**, as described in Section 3.4.2. Below all available boundary conditions are listed class wise with a short description to what they do and are intended for.

In common for all of these boundary conditions is that the program bound should take care of all specifications, no editing of the boundary conditions file should normally be necessary.

3.6.1 Connectivity Boundary Conditions

There are three different connectivity boundary conditions given in the table below.

A boundary with a periodic boundary must be connected to another boundary assigned the same boundary condition and there must be a node to node correspondence for the nodes on these boundaries. The test of the node to node correspondence is carried out in the preprocessor. Inside Edge, the fluxes on the connectivity boundaries are computed from averages of the values on two connected boundaries. For the rotation boundary conditions, the velocity vectors are rotated before averaged.

3.6.1.1 Periodic Boundary Conditions

Periodicity assumes that the boundary and the connected boundary are connected through a translational vector, and that the difference between each node on the boundaries corresponds to this vector. The vector is given as input to the program bound where also some testing is done to establish the correct sign of the vector.

3.6.1.2 Rotation Boundary Conditions

Rotation boundary condition assumes that the boundary and the connected boundary are connected by a rotation matrix. The rotation matrix is established by supplying an axis of rotation, a point on the axis and an angle. This data is supplied to the program bound where testing is done to establish the correct sign of the angle.

3.6.1.3 Propeller Boundary Conditions

The propeller boundary condition is a disk model that simulates the presence of a propeller by adding a force to the momentum and energy equations. Additional data has to be sup-

Name	Function
Weak euler	Zero normal velocity.
Weak adiabatic	No slip. Assumes zero normal derivative of the temperature.
Weak isothermal	No slip. Assumes constant wall temperature.

Table 10. Wall boundary conditions.

Name	Function
Weak euler	Zero normal velocity.

Table 11. Symmetry boundary conditions.

plied for each propeller boundary, e.g. number of propeller blades, size of propeller, radial lif force distribution etc. An example file of what data to supply is given in the file **examples/propeller_example.dat**.

3.6.2 Wall Boundary Conditions

There are three different wall boundary conditions, Euler conditions or, for viscous flows, adiabatic and isothermal conditions. The most common type for viscous flow is the adiabatic wall boundary condition. The Euler conditions can only be applied to inviscid flow.

The Euler wall condition is a weak condition, *i*.e. all variables on the wall are unknowns and the specified conditions are implied trough the flux. The two viscous boundary conditions use strong conditions for the velocity, *i*.e. the velocities are specified (usually no-slip) and are hence no longer unknowns. At an isothermal wall a weak condition is implied on the constant wall temperature.

For turbulent flows, the turbulent dependent variables are specified strongly as well. Unless otherwise stated, it is assumed that a wall resolved grid is used and that $y^+ \simeq 1$ in the first layer of nodes outside the wall. However, so called wall functions may also be used in which larger values of y^+ may be used. A wall function approach denoted "universal wall boundary conditions" is employed in which y^+ in the first layer of nodes very well can be within the boundary log layer, *i*.e. $y^+ > 30$, and where a finite velocity component is specified strongly. In principle any value of y^+ in the first layer is accepted. As the grid is refined the wall functions boundary conditions approach the standard no-slip conditions.

To use wall functions the user has to actively choose this in the program **bound** that specifies the boundary conditions for each wall boundary where wall functions shall be applied.

3.6.3 Symmetry Boundary Conditions

The symmetry boundary condition in Edge is the same condition as an Euler condition. It belongs to a separate class though to separate it from a wall boundary.

3.6.4 External Boundary Conditions

The available external boundary conditions are given in the list below. The external boundary conditions are in general weak boundary conditions, *i*.e. the variables on the boundaries

Name	Function
weak characteristic	Characteristic boundary conditions.
weak charact. vortcor	Characteristic boundary conditions with vortex correction in 2D.
weak total states	Subsonic inflow boundary condition.
strong total states pressure	Subsonic inflow boundary condition.
weak inlet	Subsonic inflow boundary condition.
weak static pressure	Subsonic outflow boundary condition.
weak extrapolation	Extrapolation of all variables.
weak free stream	All variables specified to free stream.
propulsion inlet	Propulsion inlet.
propulsion outlet	Propulsion outlet.
nacelle inlet	Nacelle inlet.
nacelle exhaust	Nacelle exhaust.
mass flow inlet	Subsonic inflow boundary condition.
mass flow outlet	Subsonic outflow boundary condition.
mass flow outlet JWS	Subsonic outflow boundary condition.
Mach outlet	Subsonic outflow boundary condition.

Table 12. External boundary conditions.

are unknowns and the specified conditions are implied trough the flux. Usually constant free stream values are used at all boundaries. Some of the boundary conditions, however, may be given local varying data by supplying the **.bedg** file with a dataset

```
N          0 x 0   ?/  boundary_data
```

as subset to the boundary in question and with local data to the boundary supplied as sub sets to this dataset boundary_data. Required data is given for each boundary condition below.

Additional dependent variables (like turbulence, species etc.) are always specified weakly (from global or local boundary data) at a local inflow or extrapolated (*i*.e. not specified) at a local outflow.

3.6.4.1 Farfield (Weak Characteristic)

This boundary conditions may be used for all external boundaries, it can handle both sub/supersonic in/outflow boundaries. It is mainly intended for external aerodynamics. It is based on a one-dimensional analysis of the Euler equations in the direction normal to the wall. Depending on the sign of the eigenvalues, the characteristics are either specified or extrapolated. See the section *Farfield Boundary using Characteristics* in the **Edge Theoretical Formulation** manual for more information.

If local varying free stream data is desired at the boundary the following minimum data is required:

```
N          0 x 0   3/  boundary_data
RF        NB x 1          density  = ...
RF        NB x 3          velocity  = ...
RF        NB x 1          pressure  = ...
```

where NB corresponds to the number of nodes on the boundary. Other dependent variables (like turbulence etc.) may be given and used if found.

3.6.4.2 Farfield with Vortex Correction (Weak Charact. Vort)

Exactly the same boundary condition as the one above with the difference that the free stream values are no longer constant but depend on the integrated lift using a so called vortex correction method. This boundary condition is intended for 2D subsonic external problems where the external boundary is located relatively close to the computed object. It gives in general an increased accuracy of the solution. Local boundary data cannot be supplied.

3.6.4.3 Total States Inlet (Weak Total States)

Total pressure, total temperature and a flow direction are specified to constant or varying conditions. The Mach number is extrapolated upstream. The boundary condition is intended as an subsonic inflow condition for internal aerodynamics. If local varying free stream data is desired at the boundary the following minimum data is required:

```
N          0 x 0   3/  boundary_data
RF        NB x 1          total_pressure  = ...
RF        NB x 1          total_temperatur  = ...
RF        NB x 3          flow_direction  = ...
```

Other dependent variables (like turbulence etc.) may be given and are used if found.

3.6.4.4 Total States Inlet (Strong total states pressure)

This boundary condition is similar to the condition above. The difference is that the static pressure is extrapolated upstream, the other variables are extrapolated. An additional differnece is the strong condition implied on all variables. Varying free stream data may be supplied as well.

3.6.4.5 Incompressible Inlet (Weak Inlet)

Density and velocity specified to constant or varying free stream conditions. Pressure is extrapolated if the flow is locally subsonic, it is specified if it is supersonic. The boundary condition is intended for an internal inflow boundary.

If local varying free stream data is desired at the boundary the following minimum data is required:

```
N          0 x 0   2/  boundary_data
RF        NB x 1          density  = ...
RF        NB x 3          velocity  = ...
```

A local pressure field may be given as well and used at supersonic conditions.

3.6.4.6 Pressure Outlet (Weak Static Pressure)

A boundary condition intended for subsonic outflows where the static pressure is specified. It may also be used at local supersonic outflow conditions, in that case the pressure is not specified (all variables extrapolated). If local varying free stream data is desired at the pressure needs to be specified:

```
N          0 x 0   1/  boundary_data
RF        NB x 1          pressure  = ...
```

3.6.4.7 Extrapolation (Weak Extrapolation)

All variables are extrapolated with this condition. Note that theoretically this boundary condition is only stable for supersonic outflow.

3.6.4.8 Free stream (Weak Free Stream)

All variables specified weakly to free stream values. Local varying free stream data may be supplied at the boundary, any variable may be spefied locally. Note that theoretically this boundary condition is only stable for supersonic inflow.

3.6.4.9 Propulsion Inlet

Boundary condition for coupling to a ram jet module. This boundary condition is not available in the official Edge distribution.

3.6.4.10 Propulsion Outlet

Boundary condition for coupling to a ram jet module. This boundary condition is not available in the official Edge distribution.

3.6.4.11 Nacelle Inlet

This boundary condition is intended for an engine nacelle infow to which the capture area ϵ is specified. Details about this boundary condition is given in Conway (2004). The boundary may be connected to the nacelle exhaust boundary.

3.6.4.12 Nacelle Exhaust

This boundary condition is intended for an engine nacelle exhaust boundary. It is required to specify pressure and temperature ratios. Details about this boundary condition is given in Conway (2004).

3.6.4.13 Mass Flow Inlet

The total temperature, the flow direction and the mass flow through the boundary are specified. The Mach number is extrapolated upstream and the total pressure adjusted to get the required mass flow. The boundary condition is intended as a subsonic inflow condition for internal aerodynamics when the mass flow is known. If local varying free stream data is desired at the boundary the following minimum data is required:

```
N          0 x 0   3/  boundary_data
RF        NB x 1          total_temperatur  = ...
RF        NB x 3          flow_direction  = ...
```

Other dependent variables (like turbulence etc.) may be given and are used if found.

3.6.4.14 Mass Flow Outlet

A boundary condition based on the weak static pressure condition and intended for subsonic outflows in internal aerodynamics. The mass flow through the boundary is specified and the static pressure adjusted so that the mass flow through the boundary matches the target. Local boundary data cannot be supplied.

3.6.4.15 Mass Flow Outlet JWS

A boundary condition based on the weak static pressure condition and intended for subsonic outflows in internal aerodynamics. The mass flow through the boundary is specified and the static pressure relaxed in time to desired value so that the mass flow through the boundary matches the target. The value of Relaxation_constant must be supplied in boundary condition file. Local boundary data cannot be supplied.

3.6.4.16 Mach Outlet

A boundary condition based on the weak static pressure and intended for subsonic outflows in internal aerodynamics. An average Mach number through the boundary is specified and the static pressure adjusted so that the Mach number at the boundary matches the target. Local boundary data can not be supplied.

3.7 Using the Vortex Generator Model

The vortex generator (VG) model simulates the presence of VGs in the solution by adding a source term to the governing equations. Thus, the VGs do not need to be physically present in the mesh.

The use of the vortex generator model is governed by the parameter VGCONFIG in the input file. VGCONFIG is the number of the VG configuration used for the computation. By default, it is zero and the VG model is not used. By "VG configuration" is meant a set of VGs. Only one set of VGs can be used per computation but the edge file (**.bedg** file) that contains the information on the VG configurations can have data for several configurations.

To import the VG geometry data in the edge file (**.bedg** file), the program **VG_identify** has to be run after the **preprocessor**. This program reads the input file where it finds the name of the file containing the definition of the VGs, CFVGNAME:

```
VG_identify inputfile.ainp
```

The format of the file CFVGNAME is described below. By looking at the geometry definition of the VGs in CFVGNAME, **VG_identify** marks all the edges where the source term should be added and update the edge file. If CFVGNAME contains data for several VG configurations, all of them are added to the edge file. **VG_identify** automatically projects the points defining the VGs on the closest wall and if VG data already exists in the edge file it is saved in a separate file.

3.7.1 The VG Geometry File **Edge.avg**

The vortex generator (VG) geometry file **Edge.avg** is an FFA-format file that contains information on the geometry of the VG configurations used by the VG model. Each VG configuration can be made of several VGs, each of them being defined by a set of points on the wall and corresponding normal and height. Each VG is approximated by a zero-thickness surface defined by the points (corners of the surfaces at the wall) and normals (sides of the surface rising from the corner points). If the VGs are straight and short only one surface is necessary but for longer VGs that follow the curvature of the wall several surfaces might be necessary to better represent the VGs.

Below, the content of the file (**ffalist**) for a configuration with one 4mm-high VG is shown, the rectangular VG being define by only two points:

```
N            0 x 0       1/  VGENDATA
N            0 x 0       1/     VGENCONFIG
N            0 x 0       3/        VGEN
DI           2 x 3                   coord  =  1.619999785
DI           2 x 3                   norm  =  0.603200000
DI           2 x 1                   height  =  4.0000E-03
```

In ASCII format, the file looks like:

```
VGENDATA          N      0          0    1
  VGENCONFIG        N       0          0    1
    VGEN              N       0          0    3
      coord             DI     3          2    0
1.61999978085664 ...
      norm              DI     3          2    0
 0.6032 0.5569 0.1478 ...
      height            DI     1          2    0
 4.000000000000000E-03 ...
```

(Note that in FFA-format the coordinates are entered as $x_1\ x_2\ \ldots x_n\ y_1\ y_2\ \ldots y_n\ z_1\ z_2\ \ldots z_n$).

3.8 Aeroelastics Options and Programs

In this section we present the programs used to prepare data for modal aeroelastic computations, including the associated file formats and describe the basic procedures for coupled and prescribed motion simulations. More details about the implementation and validation of the aeroelastic module can be found in Smith (2005).

Aeroelastic simulations are controlled from the main Edge input file by a single dataset: AEROELASTICS. The variables in this dataset are listed in Table 32. The solution option is selected by the integer variable IAEOPTas described in Table 13.

It is intended that the set of solution options will be expanded to include non modal structural models, including coupling to external structural solvers. The options series 1xx is reserved for modal solutions.

In the present version, aeroelastic solutions are always carried out as time-accurate, unsteady simulations (ITIMAQ $= 1$ in the main input file). The form of modal shape data

is further restricted to "perturbation field" (whole grid displacement field). This because on-line mesh deformation functions have not yet implemented. A further restriction is that aeroelastic solutions are limited to single processor execution.

3.8.1 Overview of the Aeroelastics "ae" Programs

Helper programs for the aeroelastics functions in Edge are denoted by the prefix "ae" and referred to as the "AE programs" (or Aeroelastics programs). There are just six such programs but the series will be extended in future versions of Edge as further aeroelastics functions are added. Several **Matlab** functions are also provided.

Each of the AE programs, including **plotmot**, have a detailed help-text (text shown when the program command is entered with `-h` or nothing). There is also a program, **ae-help**, which prints a summary of all the AE programs.

This should allow the user to run all these programs without opening this manual. The present set of AE programs, and their associated files, are listed in Table 14. Note that the program **plotmot** is a general purpose utility used with the "MOTTAB" system described in Section 3.8.7. It is included here, because this system is at present used only for Aeroelastics data.

These AE programs are used to prepare modal Aeroelastic calculation. With the 'perturbation field' method for grid deformation, the usual sequence of operations is as follows.

1. The program **aexbset** is used to set up a moveable_surface dataset defining set of boundaries which forms the moving "wetted surface" grid. This is stored in a ∼**.bset** file.

2. The program **aedbelast** (or **aedbrot** for rigid rotations) is used to generate a series of surface_movement datasets, one for each mode. These are stored in files labeled with a mode identifier ∼_**ident.bdis**

3. The script **aetribend** is used to produce, for each modeshape, a deformed mesh, a complete (**.bmsh**) mesh file conformed to the modal (**.bdis**) surface displacement.

4. The program **aeputpert** is used to calculate, for each mode, a perturbation field which is then added to a modal_shape_data dataset, stored in a **Edge.bmos** file.

The **Edge.bmos** file contains the modal perturbation fields and optionally the modal parameters (natural frequency, damping and mode description). This is the main input to Edge for any modal aeroelastic calculation.

IAEOPT	Option description
0	Rigid grids (no aeroelastics).
100	Modal Fixed Rigid.
101	Modal Prescribed Motion.
102	Modal Coupled: orthogonal modes central difference.

Table 13. External boundary conditions.

Files required	Application	Files generated
–	**ae-help**	–
Edge.bmsh	**aexbset**	∼.bset
Edge.bmsh, ∼.bset	**aedbrot**	∼.bdis
Edge.bmsh, ∼.bset ∼.amod ∼.aigi	**aedbelast**	∼_ident.bdis (or ∼.bmos)
Edge.bmsh, ∼_ident.bdis	**aetribend**	∼_ident.bmsh
∼_ident.bmsh (Edge.bmos)	**aeputpert**	Edge.bmos
Edge.bmos Edge.amot Edge.amop	**edge_***	Edge_timestep.bout Edge.bres (modal data)
Edge.amot	**plotmot**	plotmot_*.dat

Table 14. Required/generated files for Edge Aeroelastics programs.

The program **aedbelast** is used to transform structural modeshapes onto the movable- surface grid. It requires two input files, one with the modal data and one defining the interpolation scheme. For some interpolation schemes, it is possible to produce a perturbation field (and a **mos** file) directly. This is described fully in Section 3.8.4.

Depending on the required solution, the signal_table and modal_parameters datasets must be prepared, in the files **Edge.amot** and **Edge.amop** respectively. At present these files must be prepared by manual editing. The same applies to the **mod** file. This is however quite straightforward. Example files are provided in the directory **examples/ae_inputfiles**. Programs for constructing some of these input files may be added in future versions of Edge. The rest of this section contains a more detailed description of the AE programs and data files.

3.8.2 Program **aexbset** – Moving Surface Definition

The program **aexbset** is used to set up a moveable_surface dataset defining set of boundaries which forms the moving "wetted surface" grid. This data is extracted from the mesh file and written to a ∼**.bset** file (format described in Section 3.8). The moveable_surface dataset is used by the programs **aedbelast** and **aedbrotate**. In the future it may be used directly by Edge. The **aexbset** command line and arguments are as follows.

Usage

aexbset in.bmsh ut.bset moving.txt sliding.txt

Arguments

1. in.bmsh input Edge mesh file (**Edge.bmsh**)
2. ut.bset output moveable_surface dataset file
3. moving.txt input list of boundary names for movable surface
4. sliding.txt input list of boundary names for sliding planes (optional)

The two text files (arguments 3 and 4) specify the boundaries comprising the moveable surface and optionally any sliding planes (one boundary name per line). These files can

```
N          0 x 0     9/  surface_movement
L          1 x 1            brand  = "Edge 3.2 www.edge.foi.se"
L          1 x 1            title  = "AEDBELAST:  surface displacement"
L          1 x 1            parent_data  = "mdo_e05.bmsh, mdo_e05.bset..
N          0 x 0     1/     moving_surfaces
L          1 x 1               boundary_name  = "wing"
N          0 x 0     1/     sliding_planes
L          1 x 1               boundary_name  = "symmetry"
IBN     9881 x 1            nodes_moving  =  266 968 969
IBN     2512 x 1            nodes_sliding  =  1 2 3
N          0 x 0     4/     mode
I          1 x 1               identifier  =  101
L          1 x 1               description  = "1st elastic mode"
D          1 x 1               frequency_hz  =  0.8399999737739563
D          1 x 1               generalised_mass  =  10306.00000000000
DBN     9881 x 3            displacement  =  0.0000000000E+00 0.0
...
```

Figure 10. A typical moveable surface (**.bdis**) file.

most easily be constructed using the program **ffalist** with the output piped to — grep boundary_name. The boundaries must be selected visually, for example using **EnSight** or the Edge **Matlab** program **mshplot3d**.

The definition of the moving and sliding nodes follows the **Tritet** program **moving.x** which is run by the script **aetribend**. For the movable surface the mesh nodes move as prescribed by the (modal) structural coordinates. For a sliding plane (for example symmetry plane in a half-aircraft mesh) the nodes are constrained to move in-plane. The motion of all other mesh nodes is computed by the mesh deformation program (so far usually **aetribend** and **moving.x**).

3.8.2.1 The movable_surface" file ∼**.bset**

The moveable surface definition file, usually created using program **aexbset**, comprise the following datasets: moving_surfaces (and optionally sliding_planes) containing lists of boundary names; nodes_moving (and optionally nodes_sliding) containing the corresponding lists of unique node numbers. The FFA-listing of a typical ∼**.bset** file is given below.

```
N             0 x 0       7/  moveable_surface
L             1 x 1              brand  = "Edge 4.0 www.edge.foi.se"
L             1 x 1              title  = "surface data from "aexbset""
L             1 x 1              parent_data  = "mdo_e05.bmsh"
N             0 x 0       1/     moving_surfaces
L             1 x 1                 boundary_name  = "wing"
N             0 x 0       1/     sliding_planes
L             1 x 1                 boundary_name  = "symmetry"
IBN        9881 x 1              nodes_moving  =  266 968 969
IBN        2512 x 1              nodes_sliding  =  1 2 3
```

3.8.2.2 The "surface_movement" File ∼**.bdis**

The (**.bdis**) surface_movement file format is used to store surface displacements based on the moveable_surface dataset. A typical file is shown in Figure 10.

The dataset comprises a copy of the moveable_surface definition, to which is added a mode dataset containing a set of mode attributes and a displacement dataset containing the displacement vectors for all nodes contained in the moving_surfaces. The mode attributes in

this example are for an elastic mode but any attributes can be used, such as rigid body rotation parameters.

3.8.3 Program **aedbrot** – Rigid Rotation Surface Movement

The program **aedbrot** is used to create a surface_movement dataset for a rigid rotation. This is written to a **.bdis** file as described in Section 3.8. The command line and arguments are as follows.

Usage

aedbrot in.bmsh in.bset ut.bdis angle '$n_x\ n_y\ n_z$' '$x_0\ y_0\ z_0$'

Arguments

1.	in.bmsh	input mesh filename
2.	in.bset	input movaeble_surface filename
3.	ut.bdis	output surface_movement filename
4.	angle	rotation angle in, degrees (positive for right handed screw)
5.	'$n_x\ n_y\ n_z$'	rotation axis direction vector (size doesn't matter)
6.	'$x_0\ y_0\ z_0$'	reference point on the rotation axis

The first three arguments are file names and the remaining three define the rotation. The two vectors are entered as three numbers enclosed in single quotes. This applies even for 2D meshes. The rotation parameters are written to the output file as attributes of a mode dataset. These mode attributes are automatically "carried forward" in the **dis** output file as described for program **aedbelast**.

3.8.4 Program **aedbelast** – Modal Elastic Surface Movement

The program **aedbelast** is used to transform a set of mode shapes on a given structural grid into a set of surface_movement datasets defining modeshapes on the wetted surface.

Usage

aedbelast in.bmsh in.bset in.amod in.aigi [field extension option]

Arguments

1.	in.bmsh	input (**.bmsh**) mesh filename
2.	in.bset	input (**.bset**) moveable_surface filename
3.	in.amod	input (**.amod**) modes_definition filename
4.	in.aigi	input (**.aigi**) inter-grid interpolation filename

Input file 1 is a standard Edge mesh file and file 2 is produced by program **aexbset**. Input files 3 and 4 are defined in Section 3.8.4.1 and 3.8.4.2 respectively.

In normal use **aedbelast** produces a set of (**.bdis**) surface_movement files with file names of the form **modname_ident.bdis** where modname is the name of the (**.amod**) input file and ident is an integer mode identifier. Other mode attributes in the (**.amod**) file, such as frequency_hz are automatically copied into the (**.bdis**) output file.

So far, only one interpolation method is available, polyfit, a piecewise polynomial fit. With this method, the displacement can be extended out into the volume mesh to give a perturbation field. This can be used to generate a (**.bmos**) moda_shape_data file directly. To do this, these two inputs are added to the **aedbelast** command line (see ***Usage*** above) as

```
...   <filename>.bmos   <range>
```

```
N            0 x 0       4/  modes_definition
L            1 x 1              title             = "Elastic MDO wing"
IF          32 x 1              grid_idents       =  1 2 3
RF          32 x 3              str_coordinates   =  23.1191006 31.074
N            0 x 0       3/     mode_set
N            0 x 0       5/        mode
I            1 x 1                   identifier        =  101
L            1 x 1                   description       = "1st mode"
R            1 x 1                   frequency_hz      =  0.839999974
R            1 x 1                   generalised_mass  =  10306.0000
RF          32 x 3                   str_displacement  =  0.000 ...
N            0 x 0       5/        mode
I            1 x 1                   identifier        =  102
L            1 x 1                   description       = "2nd mode"
R            1 x 1                   frequency_hz      =  2.16400003
R            1 x 1                   generalised_mass  =  4552.79980
RF          32 x 3                   str_displacement  =  0.000  ...
N            0 x 0       5/        mode
I            1 x 1                   identifier        =  103
L            1 x 1                   description       = "3rd mode"
R            1 x 1                   frequency_hz      =  3.56399989
R            1 x 1                   generalised_mass  =  2457.00000
RF          32 x 3                   str_displacement  =  0.0000
...
```

Figure 11. A typical modes definition (**.amod**) file.

where <range> is the required range of the field extension, in the same units as the (**.bmsh**) grid.

Warning: This field extension function can be very slow. There is also, at present, no way to check the quality of the mesh for a given perturbation field. The recommended procedure for creating modal perturbation field data is to use the programs **aetribend** and **aeputpert**.

3.8.4.1 The "modes_definition" File ∼**.amod**

The modes_definition dataset is used to import structural modes into the Edge system. It is the Edge "native format" for this data. Conversion routines will be created, in due course, to transfer data from various structures codes to this (**.amod/.bmod**) format. The (**.amod**) file can also be created manually using the example file supplied in **examples/ae_inputfiles**.

The listing of this, three mode, example file is given in Figure 11. The dataset str_coordinates is the matrix of (x, y, z) coordinates for the selected subset of the structural grid (in this case 32 points). No element information is stored. The structural grid points are, optionally, labelled with a set of unique integer identifiers, grid_idents (default is 1,2,...).

The modal data is gathered in a mode_set and comprising mode datasets. Each mode has a number of "attributes" stored as sub datasets. The identifier is mandatory and must be a unique positive integer. The str_displacement is also mandatory and defines the modeshape.

The remaining mode attributes are optional and none of them are used by the program **aedbelast**. The parameters generalised_mass and frequency_hz may be set, or overridden, by input from the (**.amop**) modal parameter file. If the generalized_mass is absent, a default value of 1 is assumed.

All mode attributes are automatically carried forward to the (**.bmos**) modal_shape_data file.

```
N            0 x 0       3/  igi_interp_data
L            1 x 1             title             = "intergrid data"
L            1 x 1             igi_method        = "polyfit"
N            0 x 0       4/    Boxes
N            0 x 0       7/       BOX_N
RF           4 x 1                   BOX_ROOT          =  20.4300003
RF           4 x 1                   BOX_TIP           =  27.552 38.6
RF           1 x 1                   THICK             =  2.50000000
I            1 x 1                   DEGX              =  1
I            1 x 1                   DEGS              =  3
I            1 x 1                   IEDGE             =  2
I            1 x 1                   OEDGE             =  1
N            0 x 0       7/       BOX_N
RF           4 x 1                   BOX_ROOT          =  27.55 38.6
RF           4 x 1                   BOX_TIP           =  36.63 44.4
RF           1 x 1                   THICK             =  2.50000000
I            1 x 1                   DEGX              =  1
I            1 x 1                   DEGS              =  2
I            1 x 1                   IEDGE             =  0
I            1 x 1                   OEDGE             =  1
N            0 x 0       7/       BOX_N
RF           4 x 1                   BOX_ROOT          =  36.63 44.44
RF           4 x 1                   BOX_TIP           =  40.48 46.90
RF           1 x 1                   THICK             =  2.50000000
I            1 x 1                   DEGX              =  1
I            1 x 1                   DEGS              =  2
I            1 x 1                   IEDGE             =  0
I            1 x 1                   OEDGE             =  1
N            0 x 0       7/       BOX_N
RF           4 x 1                   BOX_ROOT          =  40.48 46.9
RF           4 x 1                   BOX_TIP           =  45.97 50.42
RF           1 x 1                   THICK             =  2.50000000
I            1 x 1                   DEGX              =  1
I            1 x 1                   DEGS              =  3
I            1 x 1                   IEDGE             =  0
I            1 x 1                   OEDGE             =  0
...
```

Figure 12. A typical inter-grid interpolation (**.aigi**) file.

3.8.4.2 The Inter-Grid Interpolation File ∼**.aigi**

The (**.aigi**) "inter-grid interpolation" file is used the define a transformation mapping displacements from the structural grid onto the wetted surface. This is used by the program **aedbelast** to produce a set of modeshapes on the wetted surface defined by the (**.bset**) moveable_surface file. This transformation definition data will, in the future, be used by other programs and subroutines for more general spatial coupling operations. Currently, only one interpolation method is supported, polyfit, a piecewise polynomial fit.

The (**.aigi**) input file can be prepared by manual editing using the example file provided in the directory **examples/ae_inputfiles**. A listing of this file is shown in Figure 12. The dataset igi_method is mandatory and selects the interpolation method. Everything following this dataset is specific to the interpolation method, in this case the "polyfit" method (Winzell (1996)).

In the "polyfit" method a number of boxes are defined such that the whole structure and the moving boundary is contained in the union of the boxes. In each box there will be generated a least squares fit of those displacements from **Edge.amod** in points belonging to the box, using a set of up to 64 polynomials in streamwise and spanwise (relative the box) direction and this fitted polynomial expansion is used to set the displacements on all moving boundary points within the particular box. Since the boxes can (and often should)

Option	Type	Description
-scale	real	Maximum size of surface displacement, in the (**.bmsh**) grid units
-maxit	integer	Iteration limit for mesh deformation
-delta	real	Maximum residual for mesh deformation
-D		Delete *TRITET* files at start and finish
-T		Use the **Tritet** (**moving.x**) prompt directly

Table 15. **aetribend** options.

overlap, it is always the last box that contains a particular mesh point that determines the displacements in this point.

This "polyfit" method is quite general. Its drawback is that there might occur minor discontinuities over box boundaries. It is also very important to choose the number of polynomials few enough not to introduce the overshoots that are typical of least squares fit with to high degree of polynomials.

3.8.5 Program **aetribend** – Static Mesh Deformation

The program **aetribend** is used to produce a deformed (**.bmsh**) mesh file from a base mesh file and a (**.bdis**) surface_movement file. The **aetribend** command line and arguments are as follows.

Usage

aetribend in.bmsh ut.bmsh in.bdis [Options]

Arguments

1. in.bmsh input mesh filename
2. ut.bmsh output mesh filename
3. in.bdis input surface_movement filename

This program is a Bourne shell script which runs three **Tritet** programs from the Edge mesh adaption system (**ffatri.x**, **triffa.x** and **moving.x**) and two special helper programs **aedis2tri** (see help-text) and **aetribend_tag.x**. The mesh deformation program is **moving.x** and **aetribend_tag.x** transfers mode attribute data from the (**.bdis**) surface_movement file to the output mesh file. These programs can all be used directly but **aetribend** reduces the task to a single line command.

The most important of the five options is `-scale`, which allows the user to set maximum absolute displacement size for the output deformed mesh. This is useful when the surface_movement data is a large amplitude deflection which the mesh deformation would be slow to converge or might result "negative cell volume" errors making the mesh unusable.

The `-scale` option would typically be used for "mass normalised" elastic modes where the modeshape is an unphysically large surface deflection. Data is added to the output mesh which allows the original scaling to be recovered. This is used by the program **aeputpert** to rescale the perturbation field in the (**.bmos**) modal_shape_data file, so that there is no need to adjust the generalized_mass values.

When processing a series of (**.bdis**) files, such as a set of modeshapes, with a common base mesh, it is recommended not to use the `-D` option. The program **moving.x** can then re-use some of the data required for computing the mesh displacement field (file **MOV_TRITET_FAC**). This makes the mesh deformation process significantly faster.

Options	Type	Defaults
`-ident <integer>`	Integer mode identifier	1,2,3 as added
`-scale <real>`	Scale factor for perturbation field	1.

Table 16. **aeputpert** options.

The option `-delta` sets a convergence criterion for the mesh deformation which is the maximum fractional change in the nodal displacement between successive iterations. Setting a coarser value can speed up convergence but may compromise the smoothness of the final deformed mesh.

3.8.6 Program **aeputpert** – Modal Perturbation Fields

Program **aeputpert** is used to construct the (**.bmos**) modal_shape_data file when using the perturbation field method, as described in **Edge Theoretical Formulation** manual. For this method, the modal shape data is a perturbation dataset. The format of the (**.bmos**) output file is described in Section 3.8.6.1. The **aeputpert** command line and arguments are as follows.

Usage

aeputpert in.bmsh in.bmsh iu.bmos [Options]

Arguments

1. in.bmsh input base mesh filename
2. in.bmsh input deformed mesh filename
3. iu.bmos input/output modal_shape_data filename

The program **aeputpert** automatically either *updates* and existing (**.bmos**) file or creates a new file from scratch. The perturbation field dataset is computed as the difference, $X_{\text{deformed}} - X_{\text{base}}$ where X, denotes the *coordinates* matrix of the corresponding mesh file. Any *mode* attributes stored in the *deformed* mesh file are automatically transferred to the (**.bmos**) output. The *perturbation* field is automatically rescaled using the dis_scale_factor dataset if this is present.

The manual options `-indent` and `-scale` *override* any information read from the deformed mesh file.

3.8.6.1 The "modal_shape_data" File **Edge.bmos**

The modal_shape_data (**.bmos**) file contains the entire set of *modeshape* data required for an Edge modal Aeroelastic computation. The listing of a (**.bmos**) file with three modes is shown in Figure 13.

The modal_shape_data comprises a header block followed by a series of *mode* datasets, and each of these has a set of *attributes* in the form of sub datasets. In this example, the attributes are as described for *elastic* modes in Section 3.8.4.1, with the addition of the *perturbation* field dataset. The modeshape_format dataset in the header block defines type of modeshape data. In future versions of Edge, with on-line mesh deformation, the modeshapes will be supplied as *displacements* on the moveable_surface. Currently the only supported format is 'perturbation field'.

```
N         0 x 0    7/  modal_shape_data
L         1 x 1           brand  = "Edge 3.2 www.edge.foi.se"
L         1 x 1           title  = "Modeshapes for mesh file"
L         1 x 1           modeshape_format  = "perturbation field"
I         1 x 1           n_modes  =  3
N         0 x 0    5/     mode
I         1 x 1              identifier  =  101
L         1 x 1              description  = "1st elastic mode"
D         1 x 1              frequency_hz  =  0.8399999737739563
D         1 x 1              generalised_mass  =  10306.00000000000
DF    76801 x 3              perturbation  =  0.0000000000 ...
N         0 x 0    5/     mode
I         1 x 1              identifier  =  102
L         1 x 1              description  = "2nd elastic mode"
D         1 x 1              frequency_hz  =  2.164000034332275
D         1 x 1              generalised_mass  =  4552.799804687500
DF    76801 x 3              perturbation  =  0.000000000000...
N         0 x 0    5/     mode
I         1 x 1              identifier  =  103
L         1 x 1              description  = "3rd elastic mode"
D         1 x 1              frequency_hz  =  3.563999891281128
D         1 x 1              generalised_mass  =  2457.000000000000
DF    76801 x 3              perturbation  =  0.00000000000
...
```

Figure 13. A typical modal shape data (**.bmos**) file.

3.8.7 The **mottab** Signal Tabulation System

The **mottab** system is used to input tabulated modal vector timeseries to Edge. There are several potential applications n the present code, **mottab** is only used for defining modal prescribed motion solutions. In the near future, **mottab** will be used for defining structural force inputs for coupled aeroelastic simulations and for specifying manoeuvres and similar time-dependent inputs. A **mottab** modal timeseries tabulation is always stored as signal_table dataset in a **.amot** (or **.bmot**) file.

3.8.7.1 The "signal_table" File **Edge.amot**

The **mottab** system supports two signal formats: "timeseries" and "harmonics", of which "timeseries" the most efficient and versatile. Either signal format can be used to generate on two signal types: "pulse" and "periodic" (a third type "step" will be added soon).

The **mottab** subroutine is a table-lookup function which extracts a modal coordinates and velocities from a signal_table dataset which contain data for a set of modes, each of which has a unique integer identifier. The base mode-set in Edge is defined by the (**.bmos**) modal_shape_data dataset. The signal_table controls only those modes which match an identifier in the base mode-set. The coordinates of any other "unidentified" other modes in the base mode-set are set to zero.

Examples of **mottab** signal_table files are supplied with Edge and are stored in the directory **examples/ae_inputfiles**. Time history plots from these files are stored in a subdirectory **gpics/**. The easiest way to learn how to use **mottab** is to try plotting some of these examples, using the program **plotmot**, or preferably it's **Matlab** equivalent, and tinker with some of the variables by manual editing. Similar **mottab** input files can easily be written in binary form using functions in the **FFA Matlab Toolbox**.

```
N       0 x 0    7/  signal_table
L       1 x 1           title  = "sinusoid half wave serie"
S       1 x 1           signal_format  = "timeseries"
S       1 x 1           signal_type  = "pulse"
R       1 x 1           t_step  =  4.99999989E-03
R       1 x 1           t_start  =  2.99999993E-02
R       1 x 1           amplitude_factor  =  1.00000000
N       0 x 0    2/     mode_set
N       0 x 0    2/        mode
I       1 x 1                identifier  =  101
RT   1001 x 1                timeseries  =  0.0000+00 6.28310E-03 1.2566040...
N       0 x 0    2/        mode
I       1 x 1                identifier  =  102
RT   1001 x 1                timeseries  =  0.0000000 0.000000 0.0000000 ...
...
```

Figure 14. A typical **mottab** timeseries file.

3.8.7.2 The **mottab** "timeseries" Signal Format

The listing of a typical "timeseries" signal_table is shown in Figure 14.

The signal_table dataset comprises a block of header datasets and a mode_set comprising a set of mode datasets. Each mode dataset has two mandatory attributes, the identifier, a unique, positive integer, and the timeseries, a vector of values for the modal variable at uniformly spaced intervals. The number of points in the timeseries is the same for all modes.

The dataset t_step is mandatory and defines the table time step and hence the duration of the tabulated series. The dataset signal_type is mandatory and takes the values "pulse", or "periodic". The signal_type is "periodic" then the tabulated series is repeated indefinitely to produce a periodic, wavelike signal. The effect of these two signal_type inputs is shown by these example files **time_1001_halfwave_pulse.amot** and **time_1001_halfwave_periodic.amot**.

The inputs t_start and amplitude_factor are optional, and set, respectively, a delayed onset time and a global scale factor, both of which apply to the whole mode_set.

The interpolation used in the timeseries table lookup is an overlapping three point quadratic scheme which ensures that the first derivative of the signal is continuous. For signal_type "periodic", the timeseries data is read cyclically, with periodic end conditions. For signal_type "pulse", the end conditions are such that the signal is zero outside the pulse time slice.

3.8.7.3 The **mottab** "harmonics" Signal Format

The **mottab** harmonics format represents each of the mode in the signal table as Fourier series, see (4),

$$q(t) = \frac{a_0}{2} + \sum_{k=1}^{N} a_k \cos(2\pi k f t + \phi_k) \tag{4}$$

where $q(t)$ is the modal coordinate. This format can be rather cumbersome and is included mainly to reproduce the functionality of the Euranus code. The "timeseries" format is both more versatile and faster to compute than the "harmonics" format especially for large N Fourier series.

```
N            0 x 0       6/  signal_table
L            1 x 1              title             = "appr. square-wave"
S            1 x 1              signal_format     = "harmonics"
S            1 x 1              signal_type       = "periodic"
R            1 x 1              t_start           =  5.00000007E-02
R            1 x 1              amplitude_factor  =  1.00000000
N            0 x 0       6/     mode_set
N            0 x 0       5/        mode
I            1 x 1                    identifier        =  101
R            1 x 1                    frequency_hz      =  1.00000000
R            1 x 1                    mean_position     =  0.00000000
R           64 x 1                    cos_coefficients  =  1.0  -0.33
R           64 x 1                    degrees_phases    =  180.0 0.00
N            0 x 0       5/        mode
I            1 x 1                    identifier        =  102
...
```

Figure 15. A typical **mottab** harmonics file.

The first few lines of the standard listing for one of the example files is shown in Figure 15.

As for the "timeseries" format, the signal_table dataset comprises a block of header datasets and a mode_set containing several mode datasets. The signal_type for the "harmonics" has a default setting of "periodic" but it may also be set to "puls" but if so a t_duration input is also required. The inputs t_start and amplitude_factor are optional and have the same use as for "timeseries" input.

Each mode dataset has four mandatory attributes: identifier, frequency_hz, cos_coefficients and degrees_phases, and one defaulted attribute mean_position. The identifier is, as usual, a unique positive integer. Referring to (4), the dataset names and terms match up as follows: mean_position: a_0, frequency_hz: f, cos_coefficients: a_k, and degrees_phases ϕ_k.

If the mean_position dataset is not present, the term a_0 is set to zero. The Fourier series of each mode are independent and can be of any size.

3.9 Edge Modal Coupled Solutions

3.9.1 Structural Model and Modal Parameters

Structural coupling in Edge is implemented in a simple formulation based on orthogonal structural modes. For details of this formulation, refer to the **Edge Theoretical Formulation** manual. This modal coupled simulation option is selected by setting the main input file parameter, IAEOPT $= 102$, which is located in the AEROELASTICS dataset.

The Edge-internal structural model is a set of 1D harmonic oscillators (one for each mode) which are coupled to the flow via a 'generalized force' term, the 'modal force' vector. Each modal force component is an integral of the fluid forces over the whole movable surface, weighted by the mode-shape, a surface displacement distribution defined in the **mos** input file, see Section 3.8.6.1.

The input data, defining the modal structural model and the initial conditions for the simulation, is given in the modal shape data (**mos** file Section 3.8.6.1) and the modal_parameters

dataset, the **mop** file which is described in Section 3.9.2.5. The main modal parameters are natural frequency, damping ratio and generalized mass. These may be set in the "mos" data but it is recommended to use the **mop** file input. Any value set in the **mop** file override those in the **mop** shape data file. All modal parameters and initial conditions data are echoed to the Edge log-file.

3.9.2 Coupled Simulation Procedures

3.9.2.1 General Start-up and Execution

For all coupled simulations, the initial set up procedure is the same, the program **aedbelast** is used to construct a modal_shape_data dataset (**mos** file) as described in Section 3.8.4. This contains the mode shapes as full-grid perturbation fields, plus the basic modal parameters such as frequency and generalized mass. If these parameters are not present in the "mos" data, the default values are set (frequency $= 0$, generalized mass=1). It is, however, recommended that all modal parameters are set using the **mop** file as described in Section 3.9.2.5.

In any coupled solution, the time-discretized structural equation of motion is updated within the inner iteration loop (controlled by ITMXAQ, ITMNAQ). At each update, the grid and dual mesh metrics are recomputed, which is a time-consuming process. The number of updates can be reduced using the input variable ITIN to coarsen the adjustment of the grid during the inner loop, 'pseudo time' iterations. This is echoed to standard out and to the log file under the heading "loop synchro". Inner loop convergence should be monitored using the residuals and forces on standard output.

3.9.2.2 Simulation types: Jig Shape and Design Shape

Coupled simulations may be performed in one of two scenarios, "jig shape" and "design shape", selected by the input parameter IFOFFSET.

With the default setting, IFOFFSET $= 0$, it is assumed that the geometry and structural model is "jig shape"; that is the shape applies when all loads, including structural weight, are zero. (For a wing this would be its shape when supported in the "jig" which is used to build it. This is usually very different to its shape in flight.)

With the setting IFOFFSET $= 1$, using an unsteady solution restart (INPRES $= 2$) the initial aerodynamic load is subtracted from the modal force vector. This force-offset allows a simulation to be performed using a geometry approximating to the "design shape"; the shape which the flexible wing assumes in flight.

3.9.2.3 Procedure for "Jig Shape" Simulation

The first step in a "Jig Shape" simulation is to obtain a "static" aeroelastic solution: the flow solution with the fluid loads and elastic forces in static equilibrium. Static aeroelastic solutions are obtained by setting the modal damping in the "mop" input file to an arbitrary positive value < 1. This is most easily achieved by setting the global default_damping value and leaving out the separate modal damping inputs. With heavy structural damping like this, the structural equation of motion can be integrated with a very coarse timestep (DELTAT in

the main input file). The static solution can thus be obtained with a much shorter run time than is needed for a dynamic solution.

To monitor the convergence of a "static coupled" simulation, it is usually sufficient to use the program **plotres** to display the forces and residuals. The modal forces and variables can be extracted from the **.res** file and plotted using the Edge Matlab program **plotmores**. The deformed surface shape and the surface fields can be visualized in **EnSight** using the converter program **ubody**.

For a "Jig Shape" simulation it is usually necessary to obtain the static solution for a given C_L (or other flight condition). To do this, a series of static solutions must be obtained for example as a function of incidence angle.

Once the static solution is available for the required flight condition, the dynamic simulation can be run using a restart (INPRES $= 2$ in the main input file). To restart from the static solution, however, the timestep in the initial "res" and "out" files must first be set to the value of DELTAT for the dynamic simulation. The easiest way to do this is to use the Edge Matlab program **timeset**. (One day soon a simpler procedure will be made available using and INPRES $= 1$ start up.)

On restarting from the static solution, the initial velocity for all modes must be set to zero, using the input init_velocity as described in Section 3.9.2.5. The structural excitation input can be set either via the initial conditions in the "mop" file, or by using a modal force input supplied as a "mot" timeseries input. The force input is selected via the variable IFORCED in the main input file.

The output from a modal coupled dynamic simulation is the time history data in the residuals "res" file (modal coordinates, velocity and forces) and the solution sample fields in the "out" files. The solution sampling frequency can be set using the variable IMULOU in the main input file.

3.9.2.4 Procedure for "Design Shape" Simulation

The procedure for a "design shape" simulation is simpler than for "jig shape" in the no initial static solution is required. However, some checks are necessary to ensure that the initial conditions are correct. The dynamic simulation should not be started directly (INPRES $=$ 1) from a steady state solution. Instead, the "modal rigid" option, IAEOPT $= 100$, should be used to ensure that the modal forces are constant. The dynamic solution is then started as an (INPRES $= 2$) restart from this solution.

Using the setting IFOFFSET $= 1$ offsets the modal force with the initial, steady state value. This does not, however, define an exact state of static equilibrium. For this reason it is recommended to run the dynamic simulation for a short time interval without deliberate structural excitation. On restart from the "rigid" solution the flexible structure will usually show some fluctuations. Provided that the system is in a stable, "non-flutter" state then this motion should be very small. The structure can then be excited via the "mop" initial conditions or, more physically, using a "mot" tabulated force input.

If the system is in an unstable aeroelastic state, then the transition from from the "rigid" solution may be enough to excite a "flutter" response. The initial fluctuations about the offset state would then increase in amplitude. The presence of a flutter can verified by examining the effect of increasing the structural damping.

This procedure for monitoring and analysis of a "design shape" simulation is otherwise similar to that for a "jig shape" simulation, as described above.

3.9.2.5 The "Modal_Parameters" File **Edge.amop**

A typical modal parameters **.mop** file is provided with Edge in the directory **examples/ae_inpufiles**. The FFA listing for this file is given below.

```
N  0 x 0    2/  modal_parameters
R  1 x 1           default_damping  =  1.00000005E-03
N  0 x 0    3/     mode_set
N  0 x 0    4/        mode
I  1 x 1                 identifier  =  103
R  1 x 1                 frequency_hz  =  1.10000002
R  1 x 1                 generalised_mass  =  1111.00000
R  1 x 1                 init_velocity  =  0.109999999
N  0 x 0    4/        mode
I  1 x 1                 identifier  =  101
R  1 x 1                 frequency_hz  =  2.20000005
R  1 x 1                 generalised_mass  =  2222.00000
R  1 x 1                 damping_ratio  =  1.99999996E-02
N  0 x 0    3/        mode
I  1 x 1                 identifier  =  102
R  1 x 1                 frequency_hz  =  3.29999995
R  1 x 1                 generalised_mass  =  3333.00000
```

The main dataset modal_parameters contains a single-variable dataset default_damping and a mode_set. The modal parameters are entered separately for each mode in the mode_set. In this example, there are just three modes. For each mode dataset the following input variables are available.

Dataset	Description/notes	Default
identifier	Unique integer identifier	None (mandatory input)
frequency_hz	Natural frequency	From **mos** file (or 0)
generalised_mass	Mass (a normalization factor)	From **mos** file (or 1)
damping_ratio	e.q. $= 1$ for critical damping	default_damping
init_velocity	Initial modal velocity	0
init_coordinate	Initial modal coordinate	0

The dataset default_damping is optional and if it is absent the default damping is zero.

Warning: init_velocity & init_coordinate. The "initial state" is set on restart and it there that the variables init_coordinate and init_velocity are applied. This means that if you're doing a "routine" restart, you need to save the initial **mop** file and replace it with one that does not have any of the init_ variables set. If you don't, then you will restart with the "initial" modal velocity and coordinates but with the current flow field (which is probably not what you want).

There is another possibly irksome feature in this version. Setting init_coordindate or init_velocity for any one mode automatically sets a default initial value for all modes. The default initial values are zero for both the velocity and coordinate inputs. This is sensible for most applications.

3.10 Edge Adjoint Calculation for Inviscid Flow

An example of adjoint calculation is given in **examples/rae_euler_adjoint**.

Solving an adjoint problem is an efficient way to compute gradient of a functional J that depends on the flow solution and the nodal coordinates of the computation domain, we note it $J(w, X)$, where X is the set of nodal coordinates and w the set of nodal values of the vector of conservative variables. In the current state of developement of the code Edge, only the inviscid flow equations (Euler) have an adjoint equivalent. The adjoint equations depend on J via a source term. At the time, the source terms for the following functionals are available: lift (IDCDP), drag (IDCLP), side (IDCCP), pitch (IDCMP), yaw (IDCNP) and roll (IDCRP) moments, where the variables in parenthesis describe, in the flow and adjoint input files, the reference directions used to compute the forces and moments. The source term is indicated by setting the name under the variable FUNTYP in **Adjoint.ainp**.

A brief description of the result file from an adjoint calculation is given in Section 3.4.8. In order to carry an adjoint calculation it is recommended to proceed as follows: Copy the input file **Euler.ainp** used to carry the Euler flow calculation to **Adjoint.ainp**, for example.

The file **Adjoint.ainp** will be used to carry the adjoint Euler calculation with Edge.

Since the adjoint equations are linear equations whith coefficients depending on the flow solution w it is of course required to provide the file with the Euler flow solution which name, for example Euler.bout, must be given in the variable CFIEUL in **Adjoint.ainp**.

The adjoint equations are solved if the variable ADJOPT is 1 or 2. In this version there are no more differences when running parallel or serial adjoint computations, that is explained below. If the variable ADJOPT is 0 Edge solves the flow equations. The adjoint equations that are implemented in the present version of Edge correspond to the scheme IUPWIN $= 2$ (a central scheme with a slightly different implementation than that with IUPWIN $= 0$). To be consistent with the flow calculation associated to the adjoint equations, the Euler equations should be solved using this scheme too (IUPWIN $= 2$).

The boundary conditions that are available for the adjoint equations are the counterparts of the symmetry, euler wall and farfield boundary conditions without vortex correction. Attempts to solve the adjoint equations with other boundary conditions (see the file ***.aboc** which name is given by the variable CFIBOC in both the flow and adjoint input files) will cause Edge to stop and exit with an error message:

```
ERROR IN EDGE, IN ROUTINE "ADJINI" !!!
  --- BOUNDARY TYPE "weak charact. vortcor" NOT YET IMPLEMENTED ---
```

if using a farfield boundary condition with vortex correction for example.

The adjoint calculation can be run using exactly the same options as for solving the flow (multigrid, time integration, CFL, ...).

3.10.1 Serial Computation

The required edge file and boundary file are the same as for the Euler flow calculation if ADJOPT is 1.

The output file, described in Section 3.4.8, will contain the adjoint variables. Additional calculations can be performed in Edge if ADJOPT is 2, giving the gradient of J with respect to the nodal coordinates X, in the output file that is described in Section 3.4.8.

The additional calculations (ADJOPT $= 2$) require additional data to be present in the edge file (CFIEDG is for example **Adjoint.bedg**). This is obtained by running in the usual way the preprocessor on the input file **Adjoint.ainp** (where ADJOPT $= 2$). The mesh file must be the same as the one used for the flow calculation (CFIMSH).

3.10.2 Parallel Computation

The partitionned edge files (CFIEDG) and flow solution files (CFIEUL) are required, as well as the boundary file (CFIBOC). The number of partitions (NPART in **Adjoint.ainp**) must be equal to the number of partitionned files (CFIEDG) and (CFIEUL). Option 2 of ADJOPT is now supported for parallel computations (NPART>1). This means that the output file (**CFIOUT** in **Adjoint.ainp**) contains the adjoint solution and the gradient with respect to the nodal coordinates.

Run the MPI script of Edge as usual, for example **edge_mpi_run Adjoint.ainp 10** on 10 processors. This will split the flow solution (CFIEUL) if it is not already done, and merge the adjoint solution files at the end of the calculation.

3.11 The Meshdeform Utility

The utility program **meshdeform** deforms the interior mesh given deformations (with respect to a reference mesh) of nodes on the boundaries. The directory **examples/rae_meshdeform** provides a complete example, invoked with the command

```
meshdeform meshdeform.ainp
```

The command-line argument to **meshdeform** is the name of an input file (with extension **.ainp**) that provides the file names and options needed for the **meshdeform** program. Three files, specified in the input file, are needed as input to the program:

- a reference edge file (file name in variable CFIEDG; **.bedg** format) associated with the reference mesh,
- a file containing the given boundary displacements with respect to the reference mesh (file name in variable CFIDIS; **.bdis** format), and
- a mesh file (file name in variable CFIMSH; **.bmsh** format) containing an initial guess for the deformed mesh.

The last file typically represents the same mesh as the one associated with the reference edge file, but this is not necessary: the initial-guess mesh file provides initial values for the iterative solver in **meshdeform**, so this file can be used to continue a previously aborted calculation.

The **meshdeform** program outputs a mesh file (file name in variable CFIOUM; **.bmsh** format) that contains the deformed mesh. This file will be a copy of the input file given in CFIDIS, except that the coordinates will be changed according to the mesh deformation. The coordinate vector for each mesh vertex coordinate given in the output mesh is given by

$$\boldsymbol{x}_i = \boldsymbol{x}_i^{\text{ref}} + \boldsymbol{u}_i,$$

where $\boldsymbol{x}_i^{\text{ref}}$ is the coordinate vector for corresponding vertex in the reference mesh (contained in CFIEDG). The displacements $\boldsymbol{u}_i$ are computed by minimizing the quadratic

$$\sum_{\substack{i\,j \\ \longrightarrow}} \frac{|\boldsymbol{n}_{ij}^{\text{ref}}|}{|\boldsymbol{t}_{ij}^{\text{ref}}|^p} |\boldsymbol{u}_i - \boldsymbol{u}_j|^2, \tag{5}$$

where the sum is taken over all edges in the mesh. Moreover, $\boldsymbol{t}_{ij}^{\text{ref}} = \boldsymbol{x}_j^{\text{ref}} - \boldsymbol{x}_j^{\text{ref}}$, and $\boldsymbol{n}_{ij}^{\text{ref}}$ is the dual-mesh surface vector associated with the current edge in the reference mesh.

The minimization of expression (5) can be constrained at the boundary, as specified in a boundary displacement file. The variable CFIDIS in the input file specifies the name of the boundary displacement file. Basically the same structure as described in Section 3.8 is used for the boundary displacement file. There are three ways to control the boundary points by information given in the boundary displacement file:

1. *Completely constrained motion.* The dataset nodes_moving specifies which nodes on the boundary that should be displaced, and the dataset displacement specifies corresponding displacement (all x-coordinates first, then all y-coordinates) with respect to the boundary coordinates of the reference mesh. Each node number specified in nodes_moving should be a node identified as a boundary node in the mesh file.

2. *Constrained in the local tangent plane.* A dataset sliding_planes specifies that the motion at selected boundaries should be constrained to the local tangent plane of the reference mesh. For instance, if the boundary displacement file contains the sub dataset form

   ```
   N            0 x 0       1/     sliding_planes
   L            1 x 1                 boundary_name  = "symmetry"
   ```

   the displacement vector $\boldsymbol{u}_i$ for each node on the boundary portion labeled "symmetry" will be constrained to be orthogonal to corresponding boundary normal vector $\boldsymbol{n}_i$. Note that this constraint is specified only through the name of the boundary. Thus, a dataset nodes_sliding, as in figure 10 is not needed and the presence of such a dataset will simply be ignored by **meshdeform**.

3. *Unconstrained motion.* A dataset free_surface specifies that the motion at selected boundary portions should be unconstrained in the minimization of expression (5). For instance, if the boundary displacement file contains the sub dataset form

   ```
   N            0 x 0       1/     free_surface
   L            1 x 1                 boundary_name  = "outlet_1"
   L            1 x 1                 boundary_name  = "outlet_2"
   ```

   the displacement vector $\boldsymbol{u}_i$ for each node on boundaries labeled "outlet_1" and "outlet_2" will be treated as an interior node and will thus be let free to attain the values that minimize expression (5).

The above list indicates also the hierarchy in assigning boundary condition for **meshdeform**. For instance, if the same boundary node appears in nodes_moving and in the list of nodes associated with sliding_planes, it will be treated as completely constrained (nodes_moving).

The remaining boundary nodes, that is, those nodes that are not specified in nodes_moving, or contained in sliding_planes or free_surface will be constrained to zero displacement.

The option IMDOP specifies the exponent p in expression (5). The value should be $p = 1$ or $p = 2$: $p = 2$ gives better mesh quality for difficult mesh deformations, but the convergence rate for the iterative solver is faster when $p = 1$.

There are two options for solving the discrete Laplace-like equation: using the sparse direct solver Umfpack (IMDSOL $= 1$) or a Preconditioned Conjugate Gradient (PCG) algorithm

(IMDSOL = 2). Using Umfpack requires that the Edge system is built with Umfpack as an external library (Umfpack can be downloaded free of charge), whereas the PCG algorithm is build in. The sparse direct solver is robust and fast for 2D problems. However, the memory requirements grow significantly with the size of the problem and become prohibitive for 3D (for all but very coarse meshes). Moreover the boundary condition options sliding_planes and free_surface is only implemented for the PCG solver. The memory requirements for the PCG algorithm are modest, but its convergence rate will deteriorate as the problem size grows (to alleviate this, a multigrid algorithm is under development).

There are several tests that terminate the PCG algorithm. The iterations stop when either of the following conditions have been fulfilled:

1. The maximum number of iterations ICGMIT has been reached.
2. The quotient of the square root of the current residual (5) to the square-root of the *zero-displacement* residual is less then RCGTOL. (The zero-displacement residual is the residual for zero displacements in the unknown displacements.)
3. The maximum scaled update computed at the current iteration step is less than RUPTOL. Maximum is taken over all points in the mesh, and each update is scaled with the minimum length of the edges that are associated with the current node. Setting RUPTOL to zero makes this condition inactive.

Note that the tolerances (RCGTOL and RUPTOL) and the maximum number of iterations (ICGMIT) may need to be adjusted if the PCG algorithm is used. RUPTOL= 0, and a conservatively small value of RCGTOL is used in the example. A good-enough mesh is obtained with much smaller number of iterations.

An optional flag in the input file specifies if a log file should be created. When IMDLOG=1, **meshdeform** produces a file in FFA-format with some data from the computation. No log file is produced when IMDLOG=0 or if IMDLOG is missing in the input file. The file name for the log file should be specified in variable CFILOG if IMDLOG=1. The log file is mainly useful when the iterative solver is used in order to record the residual as a function of iteration number.

References

CHEVALIER, M. 2005 Xedge v1.2, User Manual. Scientific Report FOI-D--0210--SE. Computational Physics Department, Aeronautics Division, FOI.

CONWAY, S. 2004 Implementation and Validation of an Engine Nacelle Boundary Condition in Edge 3.2. Scientific Report FOI-R--1320--SE. Computational Physics Department, Aeronautics Division, FOI.

EDGE HOMEPAGE http://www.edge.foi.se.

ELIASSON, P. 2001 EDGE, a Navier–Stokes Solver for Unstructured Grids. Scientific Report FOI-R--0298--SE. Computational Aerodynamics Department, Aeronautics Division, FOI.

HELLSTEN, A. 1998 On the solid-wall boundary condition for w in the $k-\omega$ - type turbulence modes. Scientific Report Report No B-50, Series B. Helsinki University of Technology.

MENTER, F. R. 1994 Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal* **32** (8), 1598–1605.

PENG, S. H. & ELIASSON, P. 2004 A comparison of turbulence models in prediction of flow around the DLR-F6 configuration. *AIAA Paper* **2004–4718**.

SMAGORINSKY, J. 1963 General circulation experiments with the primitive equations. *Monthly Weather Review* **91** (3), 99–164.

SMITH, J. 2005 Aeroelastic functionality in Edge, Initial Implementation and Validation. Scientific Report FOI-R--1485--SE. Computational Physics Department, Aeronautics Division, FOI.

SPALART, P. R. & ALLMARAS, S. R. 1994 A one-equation turbulence model for aerodynamic flows. *La recherche Aerospatiale* **1**, 5–21.

WALLIN, S. & JOHANSSON, A. V. 2000 An explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. *J. Fluid Mech.* **403**, 89–132.

WINZELL, B. 1996 Generation of grid perturbations for aeroelastic CFD applications with special discussion of the use within Euranus. *Tech. Rep.* TUDLF-0-96:27, Issue 1. Saab Military Aircraft.

YOSHIZAWA, A. 1986 Statistical theory for compressible shear flows, with the application to subgrid modeling. *Phys. Fluids* **29**, 2152–2164.

A Appendix – Input Variables

A.1 General Parameters for Several Programs

These file names are used in several programs.

Name	Description	Value	Default
CFIMSH	Mesh file name	*.bmsh, *.amsh	Edge.bmsh
CFIEDG	Edge file name from the preprocessor	*.bedg, *.aedg	Edge.bedg
CFIBOC	Boundary condition file	*.aboc, *.bboc	Edge.aboc
CFIRES	Residual file	*.bres, *.ares	Edge.bres
CFIOUT	Solution output file also used as initial solution for restart (INPRES=2)	*.bout, *.aout	Edge.bout
CFIMEAN	Name of time-averaged solution file	*.bout, *.aout	Edge_mean.bout
CFIINI	Initial solution file (INPRES=1)	*.bini, *.aini	Edge.bini
CFIPOST	Post processing file name	.bout	Post.bout

Table 17. File names.

The parameter NPART applies to the preprocessor, the flow solver and the programs that split and merge solution files for the parallel computations. The first six of the variables in are used by the flow solver to output lift, drag and moment during the iterations. The program 'force' uses all of the parameters to compute a complete set of forces and moments.

Name	Description	Value	Default
NPART	Number of partitions. NPART>1 specifies a parallel calculation. Used by the preprocessor and flow solver.	1	1

Table 18. Parallel calculation.

Name	Description	Value	Default
IDCLP	Direction in which the force is projected to obtain the lift		0 0 1
IDCDP	Direction in which the force is projected to obtain the drag		1 0 0
IDCMP	Direction in which the pitching moment is projected		0 1 0
IXMP	Point on the axis for all moments		0 0 0
SREF	Reference area to non-dimensionalize forces and moments		1
CREF	Reference length for the moment		1
IDCCP	Direction in which the side force is projected		0 1 0
IDCNP	Direction in which the yaw moment is projected		0 0 1
IDCRP	Direction in which the roll moment is projected		1 0 0
BREF	Reference width for the moment		1

Table 19. Integration of forces and momentums.

A.2 Preprocessor

Name	Description	Value	Default
CRDIM	Coarsening ratio for multigrid in each dimension, 2 is recommended.	> 1	2
NLEVEL	Maximum number of coarse grids produced by the preprocessor	≥ 1	5
RMSING	Removes singletons in the agglomeration (1)	0,1	1
NPLAYER	Number of prismatic layers. Used for semicoarsening in the prismatic region and, if semicoarsening is desired, should be put to the maximum number of prismatic layers in the grid.	1	1
CRATIOB	Coarsening ratio in the prismatic layer when NPLAYER>1. CRATIOB=4 is recommended. If NPLAYER=1 CRATIO apply.	> 1	4
PDUAL	Output the dual grid in EnSight 5 format to be plotted.	0,1	0
COLTYPE	Type of edge colouring 0=no action 1=Cache reordering trying all algorithms 2=Vector colouring 3=Cache reordering using saw1 algorithm 4=Cache reordering using saw2 algorithm 5=Cache reordering using bfs1 algorithm 6=Cache reordering using bfs2 algorithm 7=Cache reordering using bfs3 algorithm	0,1,2	1
NLAMREG	Number of laminar regions	0	0
LAMINAR_SETTING	Dataset containing data for all regions	No data	
LAMINAR_REGION	Dataset for each laminar region	No data	
PLANE_COO	Planes with coordinates for the laminar region		
PLANE_NORMAL	Normals of the planes with for the laminar region		
MAXLAMDIST	Maximum distance to the nearest wall for all laminar regions.	> 0	0.003

Table 20. Preprocessor variables.

A.3 Flow Solver

Name	Description	Value	Default
ITYGAS	Type of gas 1=calorically perfect gas 2=thermally perfect gas	1,2	1
GAMMA	γ	> 1	1.4
CP	C_p		1004.5
NSPEC	Number of species (ITYGAS=2)	1	1
SCHMID	Schmidt number for NSPEC>1	> 0	0.5
SCHMTU	Turbulent Schmidt number for NSPEC>1	> 0	0.9
NGTAB	Input table size for thermally perfect gas (ITYGAS=2)	≥ 1	2
IGTTAB	Table for variable temperature (ITYGAS=2)	> 0	0 2000
IGTAB	Table for variable gamma (ITYGAS=2). One dimension per specie required.	> 1	1.4 1.4
IRTAB	Table for variable gas constant RGAS (ITYGAS=2). One dimension per specie required.		287 287

Table 21. Gas related options.

Name	Description	Value	Default
INSEUL	Viscous calculation (1) Euler calculation (0)	0,1	0
ICOVIC	Constant viscosity (1) variable viscosity (0)	0,1	0
RMU	Free stream viscosity	> 0	1.7894e-5
PRREF	Prandtl number		0.72
IFULNS	Full Navier–Stokes (1) thin layer approximation (0)	0,1	0

Table 22. Viscosity related data. Parameters except INSEUL apply to viscous calculations only.

In addition to the different turbulence models available according to the table below there are specific parameters for each turbulence model that are not given here. These parameters may be found in the default input file. Note that laminar regions are specified as part of the preprocessing.

Name	Description	Value	Default
IROT	=1 for a computation in a relative frame of reference	0,1	0
OMGROT	Angular velocity and direction for the rotation		0,0,0
OMGCOO	Coordinates for a point on the axis.		0,0,0

Table 23. Computation in a relative frame of reference.

Name	Description	Value	Default
ITURB	0 for a laminar calculation 2 for a differential turbulence model 3 for DES 4 for LES	0,2,3,4	0
TURB_ MOD_ NAME	Name of the turbulence model ITURB=2 (RANS model)	'W&J EARSM + Hellsten k-o' 'Spalart-Allmaras one-eq model' 'Menter SST k-o' 'W&J DRSM + Hellsten k-o' 'std Wilcox k-o' 'Kok_TNT k-o' 'Menter BSL k-o' 'Wilcox LRN k-o' 'Rung LLR k-o' 'PDH LRN k-o' 'W&J EARSM + std k-o' 'GWJ Gen. EARSM + std k-o' 'W&J E. + Kok_TNT k-o' 'W&J E. + Menter BSL k-o' 'W&J CC-E. + std k-o' 'W&J CC-E. + Menter BSL k-o' 'W&J CC-E. + Hellsten k-o' 'W&J DRSM + std k-o' 'SSG DRSM + std k-o' 'SSG DRSM + Hellsten k-o'	
	ITURB=3 (DES model) ITURB=4 (LES model)	'Spalart-Allmaras DES model' 'Smagorinsky SGS model' 'Spalart-Allmaras SGS model' 'Yoshizawa k-eq SGS model'	
IUPWKZ	1 for upwind discretization of turbulence 0 for central + artificial dissipation discretization	0,1	1
VIS2KZ	Amount of second order artificial dissipation (IUPWKZ=0)	0	1
VIS4KZ	Amount of fourth order artificial dissipation (IUPWKZ=0)	0	0.05
TURFIX	Entropy fix for the turbulent equations (IUPWKZ=1) Scaled with spectral radius (>1) or the magnitude of the velocity (<1). 1.25 means a lowest value of 25% of spectral radius of the eigenvalue, -1.25 means 25% of velocity	<-1, >1	-1.1
IORDKZ	Order of accuracy for the upwind scheme (IUPWKZ=1)	1,2	2

Table 24. Turbulence model parameters. Note that E. is a short notation for EARSM for some models and that k-o stands for k-omega.

Name	Description	Value	Default
INPRES	0=start from free stream. 1=initial solution available, specified in CFIINI, 2=restart from previous solution specified in CFIOUT	0,1,2	0
IPOST	Post processing option, output in file name specified by CFIPOST. Several options can be combined by summation. 0 No output 1 Skin friction, heat transfer 2 Mach number, C_p and energy 4 Distance to the wall and laminar/turbulent field 8 Residual to all unknowns 16 Reynolds stresses 32 Gradients of the dependent variables 64 Flow curvature vector	≥ 0	0
PFREE	Free stream pressure	> 0	100000
TFREE	Free stream temperature	> 0	300
UFREE	Free stream velocity in x-direction		277.75
VFREE	Free stream velocity in y-direction		0.
WFREE	Free stream velocity in z-direction		0.
RMU	Free stream viscosity	> 0	1.7894e-5
TUFREE	Free stream turbulence level	> 0	0.001
VRFREE	Ratio of eddy viscosity and molecular viscosity.	> 0	1
SPMFREE	Free stream values for the species (ITYGAS=2, NSPEC>1)	0–1	1/NSPEC

Table 25. Free stream/initial solution data/post processing options.

Name	Description	Value	Default
IUPWIN	Central (0) or upwind discretization of mean flow	0,1	0
VIS2	Amount of second order artificial dissipation for central scheme	≥ 0	0.5
VIS4	Amount of fourth order artificial dissipation for central scheme	≥ 0	0.02
VIS0	Coarse grid 2nd order dissipation when simplification are used (MGSIMP=1)	≥ 0	0.15
IARDIS	Type of upwind scheme for the mean flow, 1=flux difference splitting, 2=symmetric TVD	1,2	2.
IORDER	Order of accuracy for upwind scheme for the mean flow	1,2	2.
LIMITE	Type of limiter for the mean flow, two values for the linear and non-linear fields respectively. 1=minmod, 2=van Leer, 3=superbee	2*1,2,3	2*1
IROEAV	Roe (1) or arithmetic (0) averages for the mean flow	0,1	1
ENTRFX	Entropy fix for the mean flow, two values for the linear and non-linear fields respectively. Scaled with spectral radius (> 1) or the magnitude of the velocity (< 1). 1.25 means a lowest value of 25% of spectral radius of the eigenvalue, -1.25 means 25% of velocity	2* 1, 2*-1 $\geq 1, \leq -1$	1.2, 1.05

Table 26. Spatial discretization.

Name	Description	Value	Default
NGRID	Number of grid levels	≥ 1	4
MGRSTR	Multigrid strategy, 1=V cycles, 2 W, 3 F, 4 V saw tooth cycles, 5 W saw tooth, 6 F saw tooth	1–7	1
IFULMG	Full multigrid (1) or not (0). Cannot be used with restart (INPRES=2)	0,1	0
NFMGCY	Maximum number of full multigrid cycles on coarser levels	1	500
NSWPFC	Number of sweeps on the different grid levels in coarse to fine sweep. One number for each level starting with the finest	10*0,1 ≥ 0	10*1
NSWPFC	Number of sweeps on the different grid levels in fine to coarse sweep. One number for each level starting with the finest	10*0,1 ≥ 0	10*1
MGSIMP	Multigrid simplification on coarser grids (1). Uses a first order upwind scheme on coarser grids or a simplified numerical dissipation algorithm	0,1	1
IPROLO	Type of prolongation. 0=injection, 1=account for gradients on coarser grids, 2=first order with limiters	0,1,2	0
IRESTR	Type of restriction, currently only linear	0	0
RESFMG	Convergence criterion for the full multigrid cycling. Iterations stopped when max and rms residual of the logarithm of the current residual (NRRES). Default the density residual.	< 0	-4.5

Table 27. Multigrid options.

Name	Description	Value	Default
ITMAX	Maximum number of iterations	≥ 0	1000
NSTAGE	Number of Runge–Kutta stages	≥ 1	3
IRKCO	Runge–Kutta coefficients		0.66667 0.66667 1.
ISWV	One number for each stage when to compute viscous fluxes (1) or not (0)	10*0,1	1,0,0,...
ISWS	One number for each stage when to compute source terms (1) or not (0)	10*0,1	1,0,0,...
IBOTH	1=ISWV applies to both numerical dissipation and the physical viscous dissipation, 0=ISWV applies only to physical dissipation, Numerical dissipation recalculated in all stages.	0,1	1.
CFL	CFL number	> 0	1.25
CFLVIS	Viscous CFL number. Minimum convective/viscous time step chosen. For negative values only CFL used and harmonic mean of the time steps is used	-1 or > 0	-1
CFLREF	CFL reduction factor on coarser grids. CFL and CLFVIS are reduced with where IGRID=1 is the grid level, 1 being the finest level	> 0	0.8
RESRED	Convergence criterion. Iterations stopped when max and rms residual of the logarithm of the current residual (NRRES). Default the density residual.	< 0	-5.5

Table 28. Steady state time stepping.

Name	Description	Value	Default
RSMPAR	Implicit residual smoothing parameter > 1 to be active	≥ 1	1.3
IRESMO	Type of residual smoothing. 0=standard smoothing with constant coefficients, 1=account for grid stretching	0,1	1
NSMCYC	Number of Jacobi smoothing cycles	≥ 1	2
SMCOR	Smoothing of corrections in multigrid. Defined as RSMPAR and IRESMO=0 is applied for this smoothing	≥ 1	1.5

Table 29. Residual smoothing.

Name	Description	Value	Default
ITIMAQ	Time stepping option. 0=steady state marching, 1=time accurate dual time stepping, 2=explicit time accurate	0,1,2	0
ITMAX	Number of time steps	≥ 1	1000
ISGLTS	0=use global time step DELTAT when ITIMAQ=2, 1=compute it when ITIMAQ=2	0,1	0
DELTAT	Time step when ITIMAQ 1	> 0	0.005
RESTAQ	Convergence criterion for the inner iteration in dual time stepping (ITIMAQ=1). The logarithm of the maximum norm of the chosen residual (NRRES) is used, default is the density residual.		1
ITMXAQ	Maximum number of inner iterations	> 1	100
ITMNAQ	Minimum number of inner iterations	[1, ITMXAQ]	3
BETNP1, BETN, BETNM1	The three components for the unknown in the implicit dual time stepping. Default 2nd order backward difference.		1.5, -2, 0.5
GAMNP1, GAMN	The two components for the residual in the implicit dual time stepping. Default 2nd order backward difference.		1,0
ITMEAN	Time-averaging switcher for unsteady computations (particularly DES and LES) (=1: time-averaging desired; =0: no time-averaging will be made)	0,1	0
ITERM	Time step at which time-averaging starts, function only if ITMEAN=1. NOTE: If INPRES=2 and ITMEAN=1, this indicates that time-averaging restarts from a previous run. In this case, ITERM $<$ ITER.	≥ 1	1000
NPTHIS	Number of points where time history is recorded for unsteady computations.	≥ 1	1
POINT_COOD	Coordinates of the point (x, y, z)		
POINT_INDEX	The indices of the points, 'point#' and 'bound_point' (point on a boundary surface) or 'inter_point' (interior point)		

Table 30. Unsteady time marching.

Name	Description	Value	Default
IMANEUV	0 = No manoeuvre 1 = Arbitrary manoeuvres	0,1	0
IMVDTF	0: MOVDAT format = Time, v1, v2, v3, w1, w2, w3 1: MOVDAT format = Time, v1, v2, v3, w1, w2, w3, W1, W2, W3 2: MOVDAT format = Time, v1, v2, v3, w1, w2, w3, O11, O12, O13, O21 ... O33	0,1,2	1
MOVDAT	Give description of rigid movement of grid. See format above.		

Table 31. Manoeuvre parameters.

Name	Description	Value	Default
CFIMOS	File **Edge.bmos** with mode shape data		Edge.bmos
CFIMOT	File **Edge.amot** with prescribed motion parameters		Edge.amot
CFIMOP	File Edge.amop with parameters for coupled solutions		Edge.amop
IAEOPT	0 = rigid grids 100 = modal rigid fixed grids 101 = modal prescribed motion 102 = modal coupled: orthogonal modes	0,100,101,102	0
IFORCED	0 = no forcing 1 = external modal force: MOTTAB file	0,1	0
NITIN	Structural equations updated in every NITIN inner iteration	> 0	1 1
IFOFFSET	Fluid force offset 0 = no offset (Jig Shape) 1 = offset at last timestep of "res" history > 1 = offset at numbered timestep	> 0	0

Table 32. Aeroelastic control parameters.

Name	Description	Value	Default
IPREPA	Turns on the low speed preconditioning (1)	0,1	0
RKBET2	The parameter β^2	> 0	4
RM0	The parameter M_0	> 0	1.

Table 33. Low speed preconditioning.

Name	Description	Value	Default
IWRSOL	Output solution and residual file every IWRSOL iteration	1	500
IMULOU	Outputs a unique solution (1) every IWRSOL iteration by adding the iteration number to the suffix of the output file specified in CFIOUT	0,1	0
EPS	Small number to avoid division with zero		1.0e-28
ISTOUT	Output convergence data to standard output every ISTOUT iteration	1 1	1 1 (density)
NRRES	Residual for which convergence is monitored in RESRED, RESFMG, RESTAQ		

Table 34. Miscellaneous.

A.4 Grid Adaption

Name	Description	Value	Default
CELMAX	Maximum cell size	> 0	0.7
CELMIN	Minimum cell size	> 0	0.1
CELFLO	Maximum indicator change in one cell	> 0	0.05
RMAGNI	Change number of cells maximum by this factor	> 0	4.0
IPROJ	Project new points on surface	0,1	1
YPLUS	Desired y^+ value for the first prismatic cell	> 0	1.0

Table 35. Grid adaption parameters.

A.5 Adjoint Euler

Name	Description	Value	Example
CFIEUL	Euler flow solution file		Edge_flow.bout
IUPWIN	Discretization scheme (2 for adjoint)		2 (adjoint)
ADJOPT	Type of calculation (flow or adjoint)		0 (flow), 1 or 2 (adjoint)
FUNTYP	Type of functionnal (drag, lift, side, pitch, yaw, roll)		drag

Table 36. Adjoint Euler parameters.

B Appendix – Datasets in FFA-format

B.1 Datasets for Input and Output Data

Below follows a list of some of the datasets used to define the input and output data to most of the files in both Edge and Euranus.

Dataset	Type	Dim	Size	Nsub	Explanation
grid	N	0	0	?	Grid dataset, contains the mesh
boundary_data	N	0	0	?	Dataset with boundary data
solution	N	0	0	?	Contains the solution
time_history	N	0	0	?	Time history, the dataset that contains the residuals
time	N	0	0	?	Grid solution etc. at a specific time
grid_level	N	0	0	?	Different grid levels (multi grid)
block	N	0	0	?	A block (structured (=domain) or unstructured)
free_stream_data	N	0	0	?	Free stream data

Table 37. Directory datasets.

Dataset	Type	Dim	Size	Nsub	Explanation
boundary_data	N	0	0	?	Dataset with boundary data
connectivity	IB	1	8	?	Connectivity between two boundaries
rot_symmetry	IB	1	8	?	Symmetry from rotation around an axis
trans_symmetry	IB	1	8	?	Symmetry from translation
mirror	IB	1	6	?	Mirror boundary, no flow through
pol_sing_linel	IB	1	6	?	Polar singular line, a collapsed boundary
wall	IB	1	6	?	Wall boundary
external	IB	1	6	?	External boundary
boundary_type	SI	1	1	0	Type of boundary given as data, given as Dub dataset to the boundary conditions above

Table 38. Boundary condition datasets.

The real field datasets in could also be given in double precision (8 bytes/real) with type `D**`.

Dataset	Type	Dim	Size	Nsub	Explanation
density	RFC	1	?	0	Density
velocity	RFC	3	?	0	Velocity
pressure	RFC	1	?	0	Static pressure
temperature	RFC	1	?	0	Static temperature
energy	RFC	1	?	0	Total energy
turb_kin_energy	RFC	1	?	0	Turbulent kinetic energy (k)
turb_dissipation	RFC	1	?	0	Turbulent rate of dissipation (e)
turb_time_scale	RFC	1	?	0	Turbulent time scale (t)
density_species	RFC	NSP	?	0	Density for the NSP species
vibr_energy	RFC	NVI	?	0	The NVI vibrational energies
vibr_temperature	RFC	1	?	0	The vibrational temperature

Table 39. Field datasets that might occur in the solution file.

Dataset	Type	Dim	Size	Nsub	Explanation
iteration_number	IT	1	?	0	Iteration number
Cl	RT	1	?	0	Non-dimensional lift
Cd	RT	1	?	0	Non-dimensional drag
Cm	RT	1	?	0	Non-dimensional moment
cpu_time	RT	1	?	0	CPU time
work_units	RT	1	?	0	Work units
max_heattransfer	RT	1	?	0	Maximum heat transfer
max_temperature	RT	1	?	0	Maximum temperature
total_time	RT	1	?	0	Time elapsed for time accurate calculation
inner_iterations	IT	1	?	0	# of inner iteration when time accurate (ITIMAQ $= 1$)
mode_identifiers	IT		?	0	Mode shape identifying numbers
modal_coordinate	RT		?	0	Generalized modal coordinates
modal_velocity	RT		?	0	Generalized modal velocities
modal_force_flow	RT		?	0	Generalized forces
rho_rmsresidual	RT	1	?	0	Density rms residual
rho_maxresidual	RT	1	?	0	Density max residual
rhou_rmsresidual	RT	1	?	0	x-momentum rms residual
rhou_maxresidual	RT	1	?	0	x-momentum max residual
rhov_rmsresidual	RT	1	?	0	y-momentum rms residual
rhov_maxresidual	RT	1	?	0	y-momentum max residual
rhow_rmsresidual	RT	1	?	0	z-momentum rms residual
rhow_maxresidual	RT	1	?	0	z-momentum max residual
rhoe_rmsresidual	RT	1	?	0	Energy rms residual
rhoe_maxresidual	RT	1	?	0	Energy max residual
rhok_rmsresidual	RT	1	?	0	Turbulent energy rms residual
rhok_maxresidual	RT	1	?	0	Turbulent energy max residual
rhoep_rmsresidual	RT	1	?	0	Turbulent dissipation rms residual
rhoep_maxresidual	RT	1	?	0	Turbulent dissipation max residual
rhoet_rmsresidual	RT	1	?	0	Turbulent time scale rms residual
rhoet_maxresidual	RT	1	?	0	Turbulent time scale max residual
rhos_rmsresidual	RT	NSP	?	0	Species rms residual
rhos_maxresidual	RT	NSP	?	0	Species max residual
evs_rmsresidual	RT	NVI	?	0	Electronic vibration rms residual
evs_maxresidual	RT	NVI	?	0	Electronic vibration max residual
eel_rmsresidual	RT	1	?	0	Electronic energy rms residual
eel_maxresidual	RT	1	?	0	Electronic energy max residual

Table 40. Datasets that may occur in the residual file.

FOI is an assignment-based authority under the Ministry of Defence. The core activities are research, method and technology development, as well as studies for the use of defence and security. The organization employs around 1350 people of whom around 950 are researchers. This makes FOI the largest research institute in Sweden. FOI provides its customers with leading expertise in a large number of fields such as security-policy studies and analyses in defence and security, assessment of different types of threats, systems for control and management of crises, protection against and management of hazardous substances, IT-security an the potential of new sensors.

FOI
Swedish Defence Research Agency
SE-164 90 STOCKHOLM

Tel: + 46 8 555 03000
Fax: + 46 8 555 03100

www.foi.se